Real Time Data Acquisition for Load Management

by

Sushmita Ghosh

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED:

Dr. Saifur Rahman, Chairman

Dr. C.E. Nunnally

Dr. F.C. Brockhurst

October, 1985

Blacksburg, Virginia

Real Time Data Acquisition for Load Management

by

Sushmita Ghosh

Dr. Saifur Rahman, Chairman

Electrical Engineering

(ABSTRACT)

Demand for Data Transfer between computers has increased ever since the introduction of Personal Computers (PC). Data Communicating on the Personal Computer is much more productive as it is an intelligent terminal that can connect to various hosts on the same I/O hardware circuit as well as execute processes on its own as an isolated system.

Yet, the PC on its own is useless for data communication. It requires a hardware interface circuit and software for controlling the handshaking signals and setting up communication parameters. Often the data is distorted due to noise in the line. Such transmission errors are imbedded in the data and require careful filtering.

The thesis deals with the development of a Data Acquisition system that collects real time load and weather data and stores them as historical database for use in a load forecast algorithm in a load management system. A filtering technique has been developed here that checks for transmission errors in the raw data. The microcomputers used in

this development are the IBM PC/XT and the AT&T 3B2 supermi-
cro computer.

ACKNOWLEDGEMENTS

I am extremely grateful to all the people who helped me complete this thesis. I would like to mention everybody but it is not possible to do so here.

First, I would like to thank Dr. Saifur Rahman without whose guidance I would not have been able to complete my thesis. His patience and invaluable help was a constant source of encouragement.

I would also like to thank Dr. Nunnally for his help, not only in this thesis but during the entire duration of my Master's in VPI. Without his help I would not have been able to complete my MS degree. I would also like to thank Dr. Brockhurst for agreeing to be in my committee.

Special thanks go out to for being a constant source of help. His artistic ability helped me in completing the figures and the tables.

I would like to extend my gratitude to and for their friendship, support and help in this project. My thanks also reaches out and touches AT&T for their help.

I would like to thank my sister without whose help I would probably not be here at VPI for a MS. Finally I thank my parents for all they have meant to me.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1.0 INTRODUCTION

Energy management involves efficient management of energy requirements and consumption. An user's energy cost consists of two parts:

- The cost of actual energy usage.

- The utility demand charge.

While the energy usage charge is for the energy consumed by the user and supplied by the utility, the demand charge represents the cost of maintaining and operating sufficient generation and transmission equipment to meet the power demand that the utility has agreed to supply to the user. Therefore, this demand charge is not a measure of the user's electrical energy consumption but a measure of the load connected to the utility. Hence efficient controlling of and reducing the load demand can lead to a substantial reduction in electricity costs. An efficient energy management system aims at continuously maintaining the load within a user specified demand limit as far as possible thus maintaining a flat load profile. This is achieved by measuring accumulated energy consumption from the begining of the demand interval and calculating a projected end-of-interval consumption based

on the rate of accumulation of load data and the time remaining in the interval.

1.1 DATA FLOW THROUGH THE CENTRAL COMPUTER.

The Figure 1 on page 3 gives a introductory idea about the Data Acquisition system centered around the central computer, which is the AT&T 3B2/300 supermicro. The Load data and the Weather data are collected over telephone lines. The Acquired data are then archived to form historical database files that serve as input for the load forecast algorithm.

Transmission errors due to noise in the communication line initiated the development of a software filtering technique. This filter checks for valid data coming in over the telephone lines thus maintaining a steady flow of error-free data in the data buffer.

The Termio software developed in the thesis for serial communications on the 3B2 is very useful for user independant data acquisition system. The serial I/O ports on the 3B2 are used as terminal files that can be opened, read and written to just like any other files.

1.1.1 Conclusions.

Chapter 2 describes the project set up and the hardware requirements of the Data Acquisition subsystem.

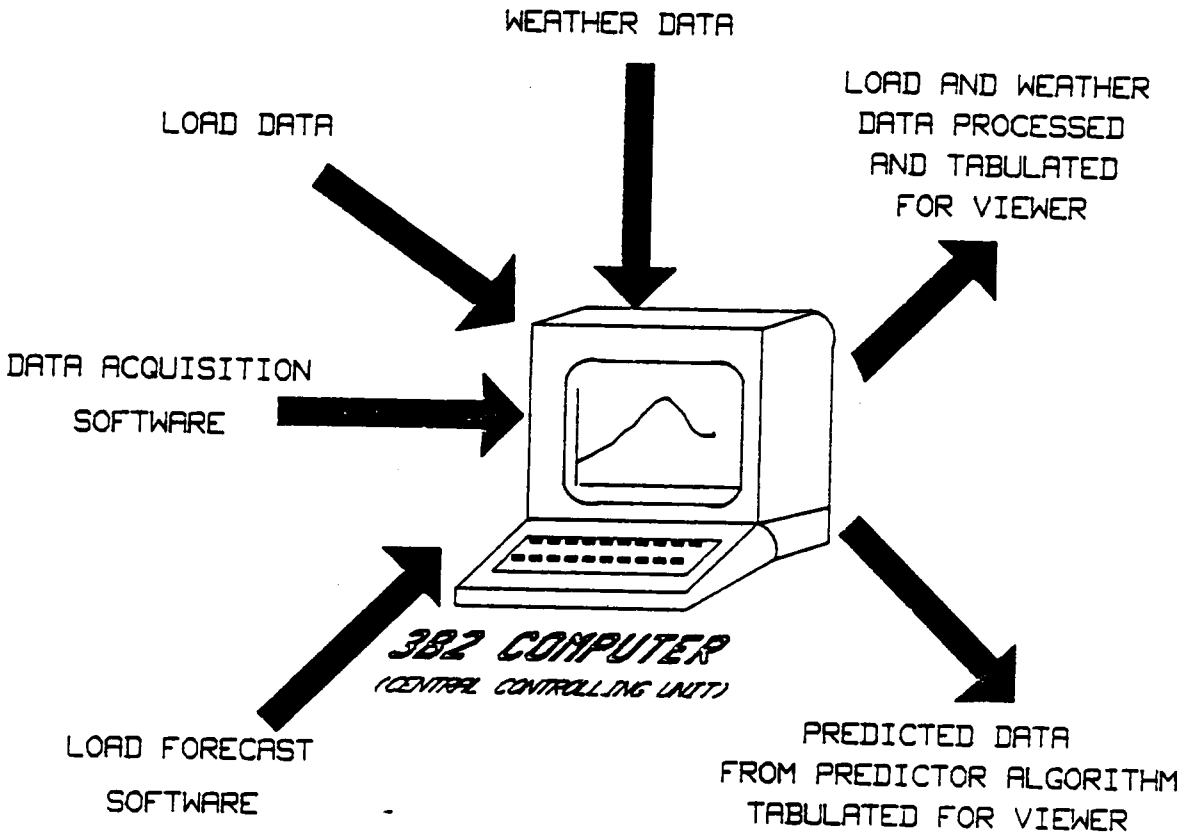

Figure 1. Data flow through the 3B2.

Error free data collection is dependant on the type of connection lines used, type of data format specified, transmission parameters, the hardware interface circuit and the data communication software that controls the hand-shaking protocol.

Chapter 3 discusses the basic communications concepts required when designing hardware and system software for data communications.

Chapter 4 deals with the Data Acquisition on the IBM PC-XT using Metrabyte modem and Penril Modem as the peripherals.

UNIX and C are very powerful tools in the world of software and they support numerous applications programs that assist in making other scientific or technical or even business application software efficient and structural.

Chapter 5 discusses the various utility programs offered by UNIX and how they have been utilized for the Data Acquisition system on the 3B2.

Chapter 6 lists the results of the Data Acquisition System on the IBM PC/XT and the 3B2 supermicro.

Appendix A lists the flowcharts of the software. Chapters 4 and 5 however, discuss the various subroutines in details.

Appendix B lists the software characteristics of the peripherals used.

2.0 FUNCTIONAL DESCRIPTION OF PROJECT.

In the introduction, the basic concept of load management has been outlined. This project is designed to provide data acquisition and processing equipment, and the necessary software to perform short term peak load forecasting for the user. If A is the power and energy supplier, all consumers of A aim to design a load management system that will predict the peak load of A. An advance notice of the oncoming A's system peak would enable the user to initiate necessary control action in order to maintain a flat load profile.

Functionally the project can be divided into two sections:

- Load Predictor Subsystem.

  The Load predictor subsystem consists mainly of software development of the load management. The Predictor software accepts and archives real time load and weather data from the Data Acquisition Subsystem. This data is then used for generating information on the size, timing and duration of the peak load. Three year historical load and weather data is used to develop the trend in the size and shape of the peak load curve. Technical details of the proposed load prediction software are not included in this thesis.

- Data Acquisition Subsystem.

  The Data acquisition subsystem generates database of historical load data and climatic data and continually updates it with new information. The real time load data is collected over a telephone line 2 & 1/2 times every second in the format '+***.***<CR>' where '*' is a numeric digit. This raw data is then filtered for transmission errors, processed and averaged over an hourly interval for subsequent storage on the historical database. The weather data is retrieved from a weather forecasting service over a switched telephone line.

  The Data Acquisition subsystem also provides Dial-in and Dial-out capability from and to remote locations through the standard switched telephone network. These sites have the capability of dialing into the central computer for retrieval of relevant data without interrupting any subsystem software execution.

2.1 PROJECT SET-UP.

Figure 2 on page 7 gives the main set-up of the project.The central computer is a multitasking multiuser supermicro which houses the complete software development of the project. It's software and hardware capabilities have been discussed in a later section. The load data is collected over a leased line and the peripheral used is a Penril modem

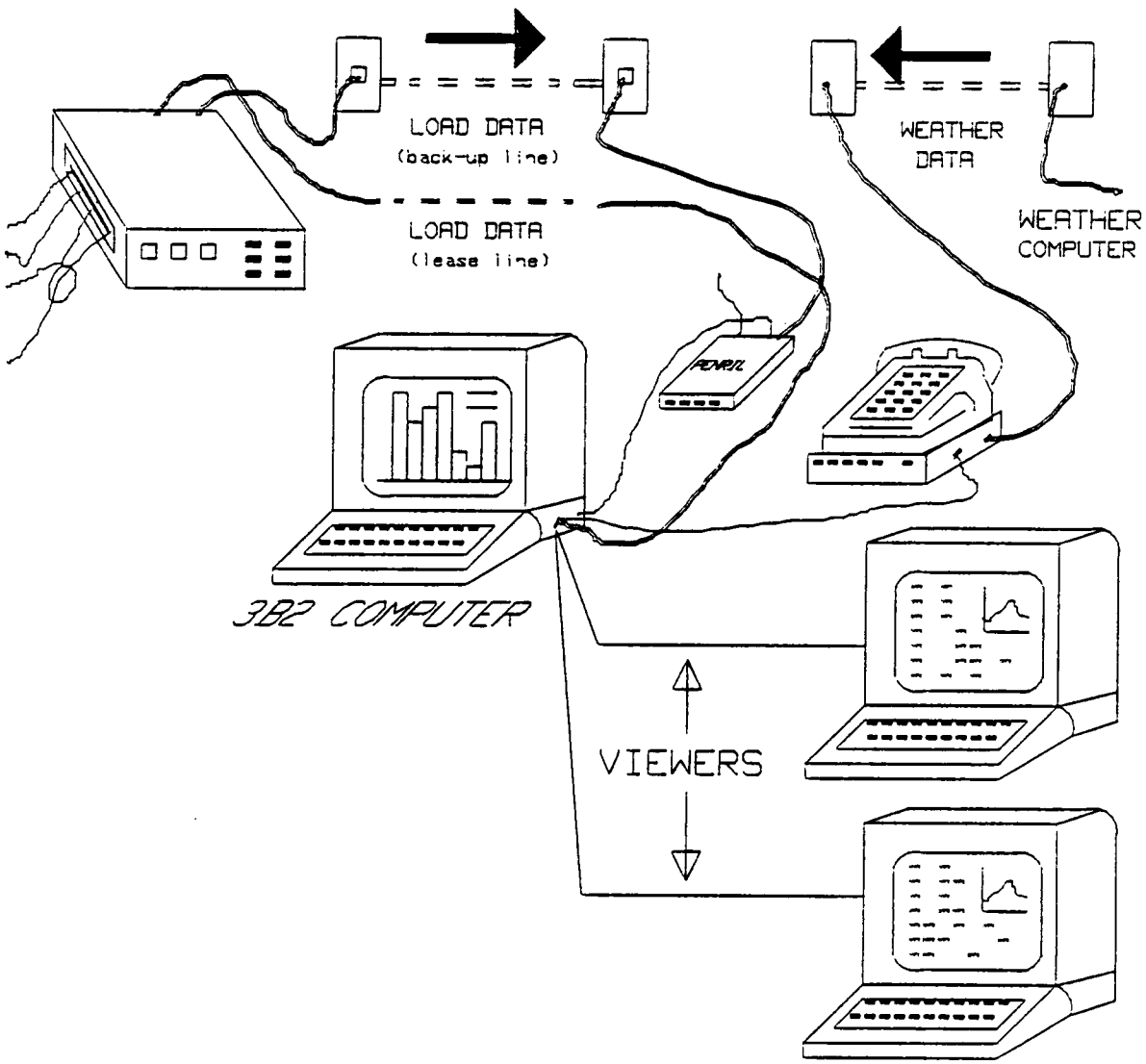

Figure 2. Project set-up.

[1]. A switched telephone line has been installed as back-up connection if the lease line failes. A second Penril modem is connected as a dedicated switched line for weather data collection (for details refer to "Data Acquisition subsystem on the 3B2/300 supermicro." on page 55). Dedicated ports are available to the viewer for retrieving any information from the computer. The prediction from the load forecasting system and the archived load and weather data is made available to the viewers in a graphics form and a tabular form.

Initially a section of the data acquisition was achieved on an IBM PC-XT [2]. Since this computer is not a multi-tasking machine, only load data was collected on COM1 I/O port of the IBM PC. This raw data was filtered, processed to produce error free data and then tabulated in a report form (see Figure 11 on page 80). The report was made available to the any user on a remote system using the secondary adapter port COM2 of the IBM PC. (for details of IBM PC I/O protocol "Data Communications for the IBM Personal computer." on page 30).

This thesis mainly discusses the Data Acquisition Subsystem part of the project including the IBM PC-XT data communications software. Later chapters are devoted to explanation of the software design techniques.

2.2 DATA ACQUISITION SUBSYSTEM.

Data Acquisition software includes data communications software, I/O file management, filtering technique of raw data, historical database management and self-diagnostic capability. The software design technique varies with the type of computer and operating environment or system. The IBM PC/XT data acquisition system was an uniprocessing one while multitasking, multiuser environment was available on the 3B2/300 supermicro.

The multitasking characteristics of the 3B2 computer and the UNIX Operating System (OS) [3] made it ideal as a central computer for the project. Before the various hardware components of the Data Acquisition subsystem is discussed two important definitions are presented here [4]. They are:

1. Uniprocessing systems.

   Uniprocessing systems are systems whose software consists of a single process or task, usually a simple loop that periodically performs some function or a set of functions. The task may either poll the inputs to find out whether any events are waiting to be serviced, or external events may interrupt the main loop. Systems in which a single main task is com bined with interrupts are called Foreground Background systems. The data acquisition software written on the IBMPC for the load data is

a Uniprocessing system. While the single task of processing the data and preparing the report for the users runs continuously in the 'background', real time data interrupts are serviced in the 'foreground' (for further details of IBMPC software explanation refer to chapter 3).

2. Multitasking systems.

A multitasking system is one in which a single computer switches attention between several sequential tasks.Multitasking differs from foreground background operation in that control is passed freely among the various tasks and there is no single background task to which control always returns. The part of a multitasking system, that manages the tasks and communication between them, is usually called the Kernel and the rest of the system is built around it. Multitasking is useful when the situation is too complicated to be dealt with by a single background task and interrupts, as in the case of this project.

In this section the hardware requirements for the data acquisition on the multiuser environment has been described in details.

2.2.1 Central Computer.

The central computer is an AT&T 3B2/300 computer [5]. This is a multiuser, desktop 32-bit supermicro that uses the WE 32000 processor and the UNIX operating system. There are two sizes of hard disks offered with the 3B2 Computer : 10 or 32 Megabyte. The Random Access Memory (RAM) expansion feature provides additional main memory to improve efficiency by increasing available buffers. This feature can expand RAM to 2 megabytes.

2.2.1.1 Input/Output Capability.

The 3B2 Computer comes with two serial ports: console port and contty port. The expanded Input/Output capability feature adds ports so one can connect more peripherals to the computer. The Input/Output feature cards each has four RS 232-C serial ports and one parallel port for a parallel printer. Four feature cards can be user installed totalling to a maximum of eighteen serial ports and four parallel ports [6].

2.2.2 Operating System

The operating system is a crucial element of a supermicro. It controls the hardware and keeps track of where in-

formation is stored. The 3B2 runs in a UNIX environment [7]. The UNIX operating system was pioneered by Ken Thompson and Dennis Ritchie at Bell Laboratories in the late 1960's. UNIX was first developed with the DEC PDP-7 in mind. Soon a version was developed that was transportable over different computers. Most of the operating system has been written in the higher-level programming language C. Today's UNIX runs on a multitude of computers namely AT&T Technologies 3B series of computers, IBM's PCs and 370 computers, DEC's PDP-11 and the VAX-11 family, Interdata's 8/32, Motorola's 68000 and Zilog's Z8000 system. The UNIX offers multitasking, multiuser environment. It consists of three basic components :

- **The scheduler:**

  This is a program that supports the multiuser capability of UNIX.

- **The shell:**

  This is the system's command interpreter and it reads the commands from the user and performs the required operations.

- **The file system:**

  This is the most fundamental element of the system and it manages the computer memory that is being used as data storage. Any form of storage on the computer is in

the form of files and the file system assists in keeping track of the files.

(For detailed information about UNIX and C see references listed later).

2.2.3 Peripherals.

Auto answer, auto answer 1200 baud modems have been used to collect data for the load management system [1]. Asynchronous communications has been established on the IBM PC/XT as well as on the 3B2 using modems and programmable I/O ports. Chapter 3 highlights the hardware and software requirements for asynchronous communications on the IBM PC-XT. Following is the description of the peripherals used with the 3B2 computer.

2.2.3.1 AT&T ACU/Modem.

This is a Bell 212a modem [1] that provides originate/answer full-duplex transmission and reception of serial binary data at 1200 and 300 bps over two wire, switched telephone lines. The modem also contains an automatic Calling Unit (ACU). The ACU is controlled by the computer connected by a EIA RS-232 interface. It also has a telephone interface with a RJ11C modular jack. The data format, operating speed

and telephone number can be stored in the modem memory cells. Appendix B lists the various specifications of the modem. Chapter 5 discusses the software designed for I/O handshaking between the 3B2 and the ACU/modem.

2.2.4 Conclusions.

Error free transmission is very essential for Data Acquisition. The next chapter highlights few important methods that help in reducing distortion due to transmission errors. In applications such as load management, where accuracy of the load forecast can contribute towards a significant reduction in energy costs, it is of utmost importance to avoid any distortion in data. Hence, it was necessary to have a software filtering technique besides using hardware precautions in this project. The filtering technique has been discussed in detail in Chapter 4.

3.0 COMMUNICATIONS CONCEPTS.

With the advent of microprocessor technology, the number of computing machines have increased and consequently, so has the need to move data between machines. This exchange of information requires a communications circuit that connects the two devices and sets up the proper hand-shaking protocol for data transfer. There are three different types of communications circuits, each characterized by it's own speed of transmission, type of transmission lines and organization [8]. They are:

1. Remote Job Entry : Some input devices such as card readers are installed at locations that are more accessible to the users than the main computer site. The user submits programs at the remote location which are transferred to the computer via data communication facility.

2. Time Sharing Device : A remote I/O terminal, such as a teletype-writer or a CRT-terminal, allows a user to enter and execute programs in an interactive fashion. Since the speed at which this happens is limited by human interaction, a number of users can normally be connected to the computer at the same time.

3. Computer-to-Computer Connection : The concept of connecting remote terminals to a computer can be extended to allow the interconnection of remotely situated computers. Such a network of computers allows transfer of data from any one computer to another.

The third type of communication circuit is the most productive as in this case the communication devices are intelligent terminals that can run foreground processes as well exchange data. Secondly, one can reduce the number of remote terminals by enabling the user to access different hosts on the same terminal. In this project the communications device is a microcomputer whose communication protocol is set up through data communications software and a smart modem. (For details refer to "Data Acquisition subsystem on the 3B2/300 supermicro." on page 55). Hence in this chapter, one communicating device will always be a personal computer (PC) while the other may be another PC or a data provider or a mainframe. Whatever the other machine is, it will be called the remote system.

3.1 TYPES OF CONNECTIONS :---

When the PC is connected to a remote system via a dialed telephone line, the line used is a switched line. These lines are fine for many types of communications but for high speed

transmission requiring a high degree of accuracy, such as load data transfers as used in this project, switched lines are noise-prone because of their unpredictable transmission line quality and susceptibility to electromagnetic interference. To avoid this lease line or a private line is used. Leased lines are like pipes with no valves nor switches and are more or less permanent and therefore not subject to the random quality of switched line connections.Yet, no line can guarantee a flawless data transfer and hence the data communication software does the necessary filtering of good data. (For more on filtering technique "Data Communications for the IBM Personal computer." on page 30)

A leased line is more expensive than switched lines but it transmits more error-free data traffic per hour and the second advantage is that one can use the same circuit to link more than one PC to the remote system. Such a line is called a multipoint line and is analogous to telephone party line.

3.2 DATA COMMUNICATION MODES.

The three types data communications modes are :

- Simplex.
- Half Duplex.
- Full Duplex.

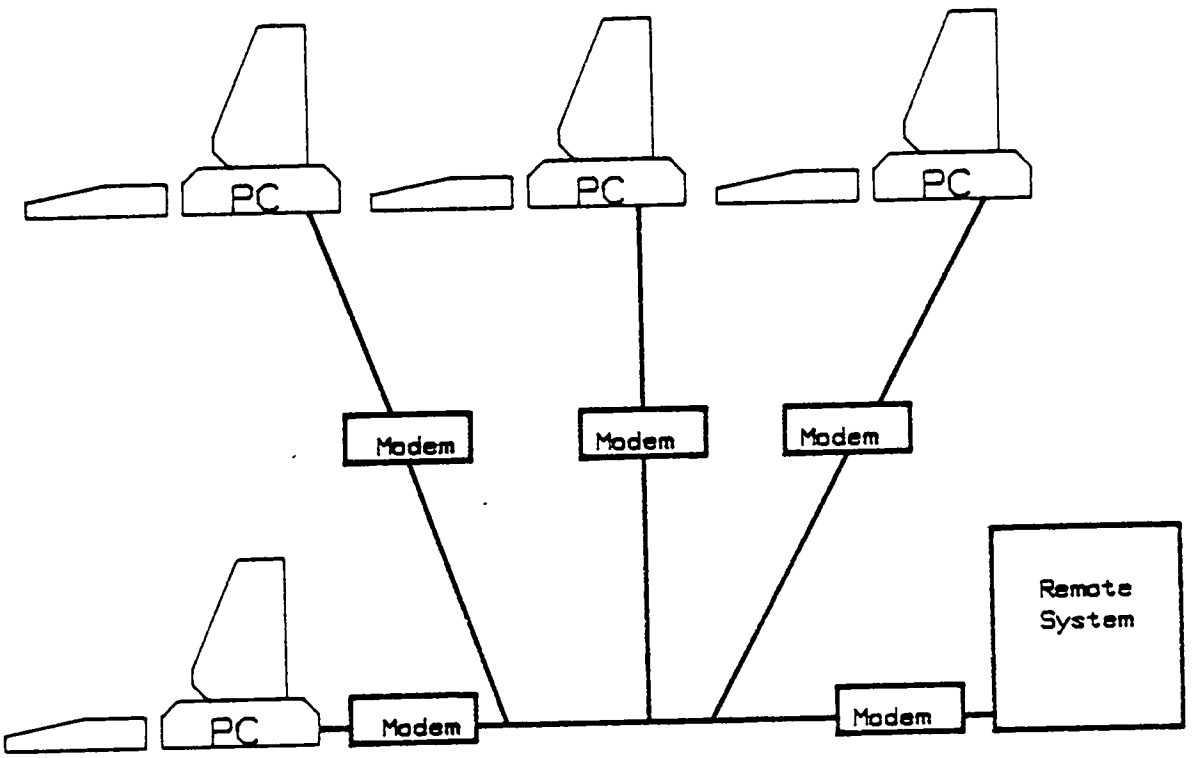

Figure 3. Multipoint Lease Line.

3.2.1 Simplex Mode.

In simplex modes, data moves in one direction only. One computer acts as transmitter and the other as the receiver.

3.2.2 Half Duplex Mode.

In half duplex modes data can move in both directions but not simultaneously. Hence in this mode,the program is required to display the data they send to the remote system since the remote is not allowed to echo data while the PC is in transmit mode.

3.2.3 Full Duplex Mode.

In full duplex mode data communication is both two-way and simultaneous, i.e, both computers on a circuit can transmit and receive simultaneously. The two individual transmission occur on separate communication channels, which combine to form a full duplex connection

In full duplex mode, the remote system frequently uses the transmission system to echo the data character the PC transmits to it. If data is the result of a PC keyboard key being pressed and a corresponding data character being transmitted, the echoed character will be readily identifiable assuming the echo transmissions were performed without

error. Since full duplex programs do not necessarily display what is being transmitted, there is no guarantee for a perfect echo.

3.3 SERIAL DATA TRANSMISSION.

For transmission of data over long distances, the cost of wires and interface circuitry required to drive starts to become significant. Moreover, as the length of the cable increases the data gets skewed. This places an upper limit on the data rate, thus eliminating the advantage of parallel data transmission. A more reasonable and common approach is to convert the parallel data to a serial format at the transmitting end and back to the parallel format at the receiving end.

Two important aspects of serial transmission of digital data that need to be considered are :

1. The nature of transmitted signal.

2. The format of transmission.

3.3.1.1 Format of transmission:

The format of transmission is of extreme importance for error free transmission of data. Data transmission errors can

occur from time to time and it is necessary to have an error checking mechanism to test whether a character is garbled or not. The following bits are included in the format of any character transmitted to guarantee error-checking :

- Asychronous start bits
- Asynchronous parity bits
- Asynchronous stop bits

Asynchronous Start Bits:: In asynchronous transmission, the line is normally idle i.e. in the MARK(1) state, (the terms MARK and SPACE are used to describe the situations corresponding to logical value 1 and 0 respectively) and due to asynchronous communication, neither the PC nor the remote system can predict exactly when the next data character will be received. Therefore,a value of binary zero, called the "start" bit is transmitted immediately before any actual data bits are transmitted. Thus when the PC is ready to transmit the character, it automatically precedes the character with a start bit,which signals the remote system that a data is being transmitted.

Asynchronous Parity Bits:: If data communications were 100 percent reliable, there would be no need to send any periph-

eral signals. But since telecommunications are not completely reliable, the receiving system needs at least a rudimentary method of checking whether it correctly received what was sent. This is provided by inserting an optional check bit called the parity bit, immediately after sending the last data character bit.

The parity is an eighth bit that can be transmitted with each seven bit character. It increases the chance of detecting data that is garbled in transmission. In contrast to the start bit, the inclusion of the parity bit is optional. Once the type of parity generation scheme is chosen, both the PC and the remote system must observe the same parity bit convention during any transmission.

Parity Generation Schemes:: Parity can be specified in five ways for use during a data communications session:

1. NONE: This method specifies that no parity should be generated for transmitted characters and that none should be expected to accompany the arriving characters. The parity bit position is used for a data bit.

2. MARK: This method specifies that parity bit will always be a binary one. Thus the eighth bit is a MARK.

3. SPACE: This method specifies that the parity bit will always be a binary zero. Important to remember that the parity is only a position filler and not included as actual data bit.

4. EVEN: This most common method specifies that the value of the parity bit will depend on the specific character it accompanies and will be a binary value that will make the total number of parity and data bits having an even number of binary ones. In other words, the parity-generating scheme counts the "one" bits in a character, and if the number is odd, the parity bit will be a binary "one" to make it even; if the original count is even, the parity bit is binary "zero" to keep it even. However, this check is not really fool-proof as transmission error that change any two, four or six bits will not be detected.

5. ODD: This method specifies that value of the parity bit will depend on the specific character it accompanies and will be a binary value that will make the total number of parity and data bits having an odd number of binary ones.

ASCII CHARACTER	HEXADECIMAL VALUE	BINARY VALUE	PARITY METHOD				
			MARK	SPACE	EVEN	ODD	NONE
A	41	10000001	1	0	0 (2)	1 (3)	n/a
B	42	10000010	1	0	0 (2)	1 (3)	n/a
C	43	10000011	1	0	1 (4)	0 (3)	n/a
a	61	11000001	1	0	1 (4)	0 (3)	n/a
b	62	11000010	1	0	1 (4)	0 (3)	n/a
c	63	11000011	1	0	0 (4)	1 (5)	n/a

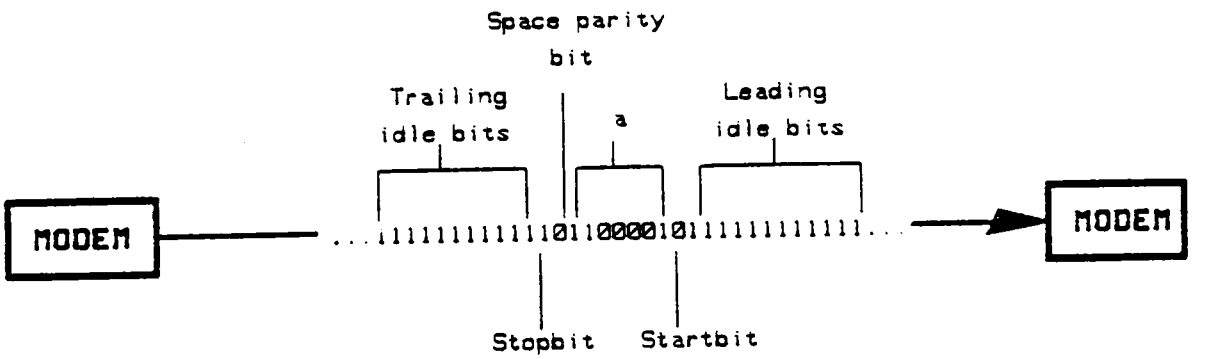

Figure 4. Five Parity Methods For Six Ascii characters.

Asynchronous Stop Bits:: Sometimes transmission errors occur such that it indicates to the PC that a character is arriving when actually none is. This error is avoided by terminating the character and the optional parity bit with one or two bits used in much the same way as a period at the end of the sentence. These bits are called "stop" bits and have a binary value of one. A data character can be accompanied by one or two stop bits. The stop bits guarantee a transition from a binary one to a binary zero when a data arrives so that the next character's "start" bit can be recognized immediately even if two characters come in quick succession.

The advantage of asynchronous transmission is, of course, that character can be sent at random intervals. A secondary advantage is that if a bit is scrambled due to noise, only the character that contains the scrambled bit is affected. Finally, since the start of each character is explicitly marked, small difference in clock speed between transmitter and receiver can be compensated for. Figure 4 on page 24 shows the configuration of bit stream for binary 'a'.

3.3.2 Serial Terminal Interfaces.

A large family of serial interfaces developed out of the use of serial teletype-writers as computer terminals. Since many computer systems have terminals (including almost all

general purpose microcomputer system), computer terminal interfaces have become very widespread, and are now used for computer to computer communications and for communications with an immense variety of peripherals.

3.3.2.1 RS 232-C Interface.

The most widely used standard for serial data transmission is the RS 232-C standard, which uses a 5-12V to indicate a one and -5 to -12 for a zero. The threshold for the receiver are 3 and -3 volts, giving reasonable noise margins. RS 232-C is usable for cable lengths up to 50 ft. and speeds up to 20K bits per second.

The standard also specifies a connector and pinouts for the data signals. A RS 232-C serial link is often the only way of getting information from one computer to another. RS 232-C is intended to link a computer or terminal(DTE or Data Terminal Equipment) to a modem(DCE or a Data Communication Equipment).

3.3.2.2 Modems.

For transmission over longer distances, an A.C signal is required. A MODEM (MOdulator-DEModulator) forms a A.C transmission medium for interface between digital and analog

systems. The modems used for the data acquisition system on the IBM PC/XT are:

- INFO-MATE 212PC 110/300/1200 bps modem that plugs directly into any one of the IBM PC expansion slots [9]. (For details refer to Chapter 4).

- Penril modem is a AT&T 300/1200 bps modem [1]. It has been discussed in detail in Chapter 2.

The Penril modem is used for the data acquisition on the 3B2 supermicro. There are other serial interfaces such as LAN coaxial cables, current loops ES423 and RS 422 and others [4], but details of these have been omitted as they were not used here.

3.3.3 Transmission Speed.

The speed of serial transmission is the data rate or bit rate measured in bits per second. The term 'BAUD' is commonly used instead of bits per second (bps). The most common data rates in use today are 300 and 1200 baud for telephone connections and 9600 and 19200 baud for direct links.

For the data acquisition system on the PC, 1200 baud was used for data collection and 300 baud was the rate at which the report was sent out via COM2. For the data acquisition

system on the 3B2, load was collected at 1200 baud, weather was collected at 300 baud and outside callers to the 3B2 ports were at 1200 baud. The baud rate has to be the same for both ends in any data communication link to avoid any distortion in transmission.

3.3.4 Conclusion.

This chapter has outlined the basic concepts of communication and a few strategies for avoiding transmission errors which is very necessary in any Data Acquisition system. Discussion about the technical details of any microcomputer has been deliberately omitted in this chapter as Chapters 2, 3 and 5 deal in detail with the computers involved and the software in them. Before concluding however, there is a interesting information to be noted about the role of a personal computer in data communication. With the need to move data between computers, the demand for personal computer as a communication terminal has soared up due to the following three needs:

1. P.C users are finding limitations in either processing or data availability and therefore link up to expand their personal computer.

2. Computer users often need to access different hosts and a personal computer eliminates the need to have more than a single terminal around.

3. Intelligent terminals as a personal computer are more productive than traditional dumb terminals.

Furthermore, multitasking microcomputer or multitasking operating system are becoming very popular, as they assist in enhancing the existing capabilities of the microcomputer, thus making it inevitable in any technical or business application programs.

4.0 DATA COMMUNICATIONS FOR THE IBM PERSONAL COMPUTER.

Communications or data transfer to and from any personal computer requires a hardware interface circuit for the hand-shaking signals and data communication software for user access of the I/O ports.

The IBM PC has a Asynchronous Communication Adapter card that provides part of the hardware interface circuit [2]. The heart of the adapter is a INS8250 LSI chip which is fully programmable and controls all asynchronous communications on the PC. This adapter has a fully prioritized interrupt system that controls the transmit, receive, error, line status and data set interrupts. It provides an EIA RS-232C type interface.

4.1 INTERFACE DESCRIPTION.

The RS 232-C interface provides current and voltage interface. The voltage interface is a serial interface and supports certain data and control signals as listed below:

- Pin 2 Transmitted data (**TD**).
- pin 3 Received data (**RD**).
- Pin 4 Request To Send (**RTS**).
- Pin 5 Clear To Send (**CTS**).

- Pin 6 Data Set Ready (**DSR**).
- Pin 8 Carrier Detect (**CD**).
- Pin 20 Data Terminal Ready (**DTR**).

The adapter converts these signals to and from TTL levels to EIA voltage levels. These signals are sampled or generated by the communication control chip. Then they are ready to be sensed by the system software to determine the state of the interface or peripheral device. When these EIA level voltages appear at the RS 232-C interface of the adapter, they are transferred to a modem (MOdulator-DEModulator) device that completes the hardware interface circuit requirement for communications.

Though the asynchronous adapter card and any smart modem could efficiently deal with data transfer on the PC, a Metrabyte 110/300/1200 bps Intelligent Modem card was used for the data acquisition. This Modem card plugs into any of the IBM PC's expansion slots and has a RS232-C interface as well as a RJ-11 connection built-in and hence no external modem was required. Both the serial I/O ports available on the IBM PC/XT were utilized in the Data Acquisition system. The raw data was collected on the COM1 port, while the COM2 was used for transmitting archived data to callers. Though two Metrabyte modems were plugged into the IBM PC/XT, only one was programmed to control the I/O handling of the serial port. The second one was used only as a RS232-C interface,

the I/O handling of it's serial port was handled by Penril modem.

The heart of the Metrabyte modem is also an INS8250 chip and so the programming considerations of the 8250 for communications is the same irrespective of the modem used.

4.2 PROGRAMMING CONSIDERATIONS USING THE MODEM CARD.

This modem contains all hardware required for communications protocol and is also equipped with data communication software that programs the personal computer to act as an intelligent, menu driven, communications terminal. It accepts 14 different serial commands such as Dial, Answer, End to automatically dial, answer and hang-up calls. [9]

Since the software is menu driven, it requires keyboard input from user as data for command execution as well as for setting of communication parameters such as parity, speed etc. Such software is of limited use in real_time application programs where independence of user interaction is an essential requirement. Suppose bad weather disconnects the telephone line or a power failure occurs which results in termination of the data acquisition program. On such occasions, it is essential to have automatic restart of program without user interaction so that when the line is back up again or the system is powered again the software does not

have to wait for keyboard commands from user. The software written for PC data communications in this project was made user independant, i.e, the handshaking protocol was all within the software Before details of the software design technique is discussed, a few programming considerations of the 8250 chip are described.

4.2.1 Programming the 8250.

In the I/O address map of the 8088 system processor,the addresses hex 3f8 to hex 3ff are reserved for the primary adapter card. The secondary adapter is accessed at locations hex 2f8 to hex 2ff [2]. The adapter addresses are selected by plugging the programmed shunt moduler with the locater dots up or down. (See to Figure 5 on page 34).

4.2.1.1 Output signals of the 8250.

Following is the description of four important output signals of the 8250 chip that play a major role in data communication [2]

- Data Terminal Ready (DTR), Pin 33: When low, it informs the modem or data set that the 8250 is ready to communi-

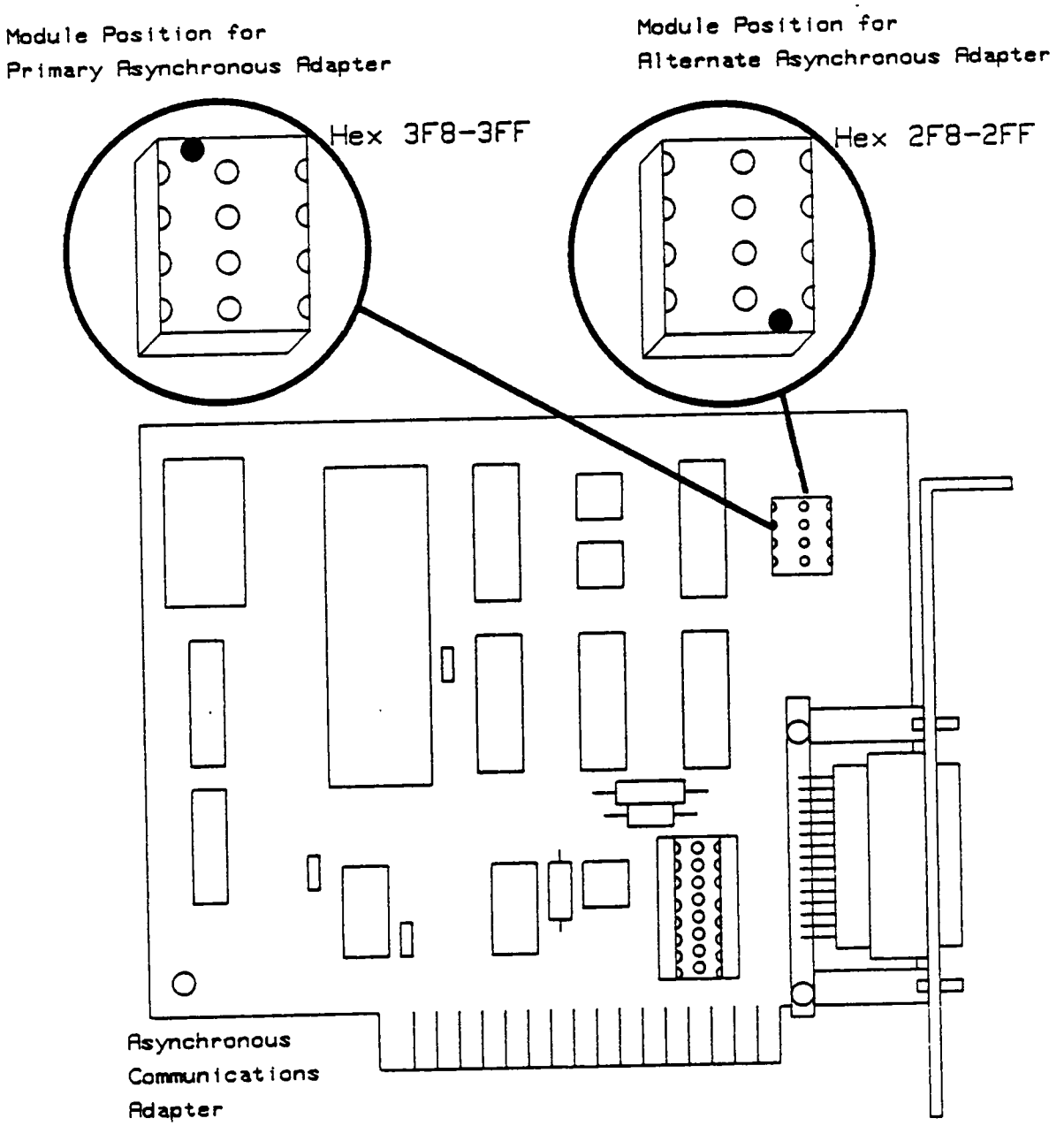

Figure 5. Block Diagram of the Adapter.

cate. The DTR output signal can be set to an active low by programming bit 0 (DTR) of the modem control register to a high level(+2.4 V dc).

- Request To Send (RTS), Pin 32: When low, it informs the modem or data set that the 8250 is ready to transmit data. The RTS output signal can be set to an active low by programming bit 1 (RTS) of the modem control register.

- Output 1 (OUT1), Pin31 : This can be set to an active low by programming bit 2 (OUT1) of modem control register. This is an user designated output and the Metrabyte Modem card uses this signal to enable the serial port COM1's connection to switch between the modem and the RS 232-C connector under software control.

- Output 2 (OUT2), Pin31 : This can be set to an active low by programming bit 2 (OUT1) of modem control register. This is an user designated output and the Metrabyte Modem card uses this signal to enable the serial port COM2's connection to switch between the modem and the RS 232-C connector under software control.

4.2.1.2 Functional Description of 8250 registers.

The IN8250 has a number of accessible registers that may be accessed or controlled by the user through the processor. The data communications software enables the user to access the numerous registers of the 8250 by selecting the particular address from the 8088 I/O map [2]. Hence programming the 8250 is achieved by selecting the address of the I/O registers and writing data out to the adapter card. (refer to Figure 6 on page 37) The C-86 compiler used for the data acquisition on the IBM PC-XT supports few useful I/O macros that enable the user to write to and read from any I/O device that has a valid address in the IBM PC I/O address map [2]. The most commonly used in this software are:

- INPORTB(I/O address) for reading devices.
- OUTPORTB(I/O address,data) for writing to devices.

The various registers define the different modes of operation and address bits A0,A1 and A2 select the registers. (refer to Figure 6 on page 37)

1. Line-Control Register.

   The software specifies the format of the asynchronous data communications exchange through the line-control register. In addition to controlling the format,

I/O Decode (in Hex)		Register Selected	DLAB State
Primary Adapter	Alternate Adapter		
3F8	2F8	TX Buffer	DLAB=0(Write)
3F8	2F8	RX Buffer	DLAB=0(Read)
3F8	2F8	Divisor Latch LSB	DLAB=1
3F9	2F9	Divisor Latch MSB	DLAB=1
3F9	2F9	Interrupt Enable Register	
3FA	2FA	Interrupt Identification Registers	
3FB	2FB	Line Control Register	
3FC	2FC	Modem Control Register	
3FD	2FD	Line Status Register	
3FE	2FE	Modem Status Register	

I/O DECODES

Hex Address 3F8 to 3FF and 2F8 to 2FF											
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	DLAB	Register
1	1/0	1	1	1	1	1	X	X	X		
							0	0	0	0	Receive Buffer (read) Transmit Holding Reg. (write)
							0	0	1	0	Interrupt Enable
							0	1	0	X	Interrupt Identification
							0	1	1	X	Line Control
							1	0	0	X	Modem Control
							1	0	1	X	Line Status
							1	1	0	X	Modem Status
							1	1	1	X	None
							0	0	0	1	Divisor Latch (LSB)
							0	0	1	1	Divisor Latch (MSB)

Figure 6. Table for I/O addressing of 8250 registers.

the programmer may retrieve the contents of the line-control register for inspection. This is achieved by using the macros INPORTB or OUTPORTB mentioned above. The C-86 compiler supports a dedicated macro COM_RST for initialization of I/O ports for communications [10]. The input requirements of the macro is:

a. Port address to specify COM1 or COM2.
b. Baud rate of transmission.
c. Parity used.
d. Stop Bits.
e. Bits per character transmitted.

This macro is useful when the baud rate is 1200 bps or higher. For lower baud rates the INT 14 of the ROM BIOS routines can be used. The COM_RST is a high level macro that programmes the line-control register of the 8250 with the parameters specified as it's input variables.

2. Interrupt Enable Register.

This eight-bit register enables four types of interrupt of the 8250 namely:

a. Data available interrupt.
b. Transmit holding register empty interrupt.
c. Recieve line status interrupt.
d. Modem status interrupt.

Since data is being collected on the COM1 port by polling, the data available interrupt and the recieve line status interrupt are enabled by setting the corresponding bits in the Interrupt Enable Register whose I/O address is hex 3F9.

3. Line status register.

This eight-bit register provides status information on the processor concerning the data transfer. The most important bit of this register is the zero-th bit which is the receiver data ready indicator. This bit is set to a logical 1 whenever a complete incoming character has been recieved and transferred into the reciever buffer register. The line status register is polled for data arrival in the COM1 subroutine COMINT by checking for the bit 0.

4. Modem control register.

This eight bit register is by far the most important of the registers for communications as it controls the interface of the processor with the modem or the data set. It's programming considerations are explained in an earlier section that discusses the output signals of the IN8250. The COM2 port of the PC is dedicated for viewers to see the load report and this port is controlled the Metrabyte modem. Hence the bit 2 of the modem control

register that controls the COM2 (I/O address hex 2FC) is reset. The COM1 port of the PC collects data using an external Penril modem and hence the serial port is connected directly to the Penril thus bypassing the Metrabyte modem. This is achieved by setting bit 2 of the modem control register that controls the COM1 (I/O address hex 3FC). This bit controls the OUT1 output signal of the IN8250 [2].

4.2.2 Metrabyte Modem Option Switches.

As mentioned earlier in the chapter, two modems were used, one on each serial I/O port.
These INFO-MATE 212PC modems have hardware switches that can be set by the user before plugging it in. Figure 7 on page 41 lists the various settings options of these switches [9]. These switch settings for option 1 decide the serial port address to be assigned to that modem, thus giving the option of having two I/O ports COM1 and COM2. One of the modems was assigned COM2 port addressess and was software controlled. This port processed the incoming calls and transmitted the data. The modem assigned COM1 was only used as a RS232-C interface and was used for collection of raw data.

The switch setting for option 4 is important as it decides on control of the I/O port. With the switch W9 'ON' and switch W10 'OFF', the I/O control of the port can be chosen

NO.	OPTION	SETTING	SWITCH POSITION
1.	SERIAL PORT ADDRESS	COM1: (3F8 hex)	W1 W2 W6 W7 W3 W12 ON OFF ON OFF ON OFF
		COM2: (2F8hex)	OFF ON OFF ON OFF ON
2.	SERIAL DATA FORMAT	9 data bits	W4-ON
		8 data bits or 7 bits with 2 stop bits	W4-OFF
3.	MODEM CONTROL LINE OPERATION (CTS, DSR, and DCD)	follows DTR	W5-ON
		follows NORMAL RS 232C format	W5-OFF
4.	MODEM/ AUXILARY RS 232C PORT SELECTION	modem	W9 W10 OFF OFF
		RS 232C	OFF ON
		modem or RS 232C select by PC software	ON OFF

Figure 7. INFO-MATE modem option switch settings.

by software, by programming bit 2 of the MODEM CONTROL Register (see previous section). This setting was used on the modem on COM2. This modem was controlled by software and it dealt with all I/O handling required on COM2. The modem on COM1 was used only as a RS232-C interface with switch settings for option 4 being ' **W9 OFF and W10 ON** '. (refer to Figure 7 on page 41).

4.3 DATA ACQUISITION SOFTWARE.

Since the heart of the Metrabyte Modem is a IN8250 chip, programming the modem is basically programming the IN8250 registers. The various commands sent to the modem for receiving data on COM2, sending data on COM2 and hanging up telephone connection after termination of report are discussed in details in this section.

The Figure 8 on page 43 shows the set-up used on the IM PC/XT for data collection. The switched telephone line for callers was connected to COM2 I/O port. The COM1 has a leased line connected to it. If the lease line fails, the COM1 is then connected to a switched line for data collection. This is the back-up line and is used only when necessary.

The Penril modem used on COM1 for load data collection is an AT&T modem and it's hardware characteristics have been described in "Peripherals." on page 13. The commands and responses of this modem are discussed in Appendix B.

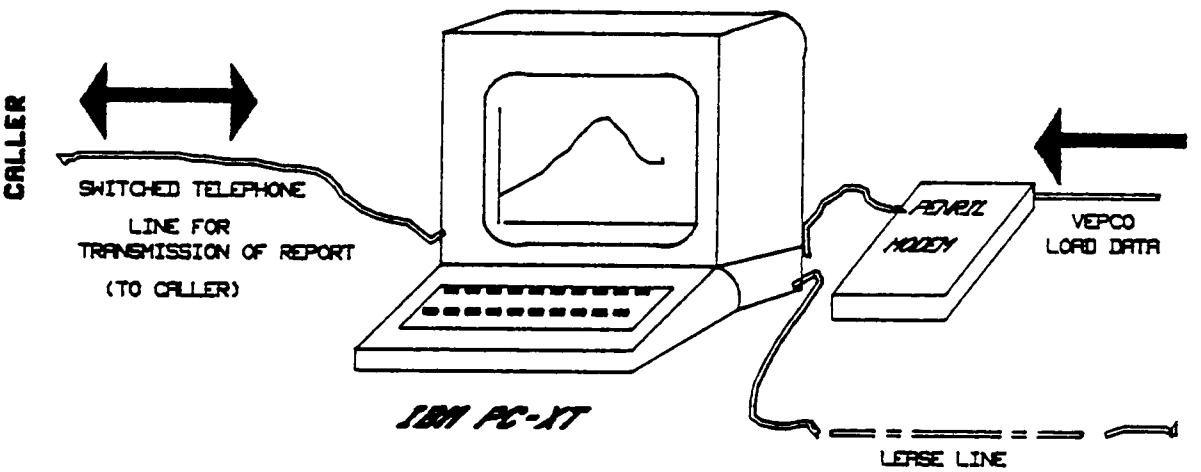

Figure 8. The Data Acquisition setup on the IBM PC/XT.

This section discusses the I/O handshaking protocol, the COM1 and COM2 interrupt handling routine and the main processing code of the Data Acquisition Software on the IBM PC-XT.

Since the software is a uniprocessing one (refer to"Data Acquisition Subsystem." on page 9), it has a single major loop that handles all tasks. This section is divided into subsections according to the tasks handled which include file management of load data, processing of data into a report and polling of COM2 for callers, collection of data from COM1 using interrupt routine, filtering of raw data for transmission errors and data update on console terminal using cursor control.

4.3.1 Subroutine COM_INT

This routine handles the polling of the line-status register for available data on COM1, filters the raw data for transmission errors and updates the ring buffer HR_BUF 2 1/2 times every second. The load data is of the format '+***.***' where each * is a numerical digit from 0 to 9. The termination of each data value is indicated by a carriage return <CR>, ASCII value Hex 0D.

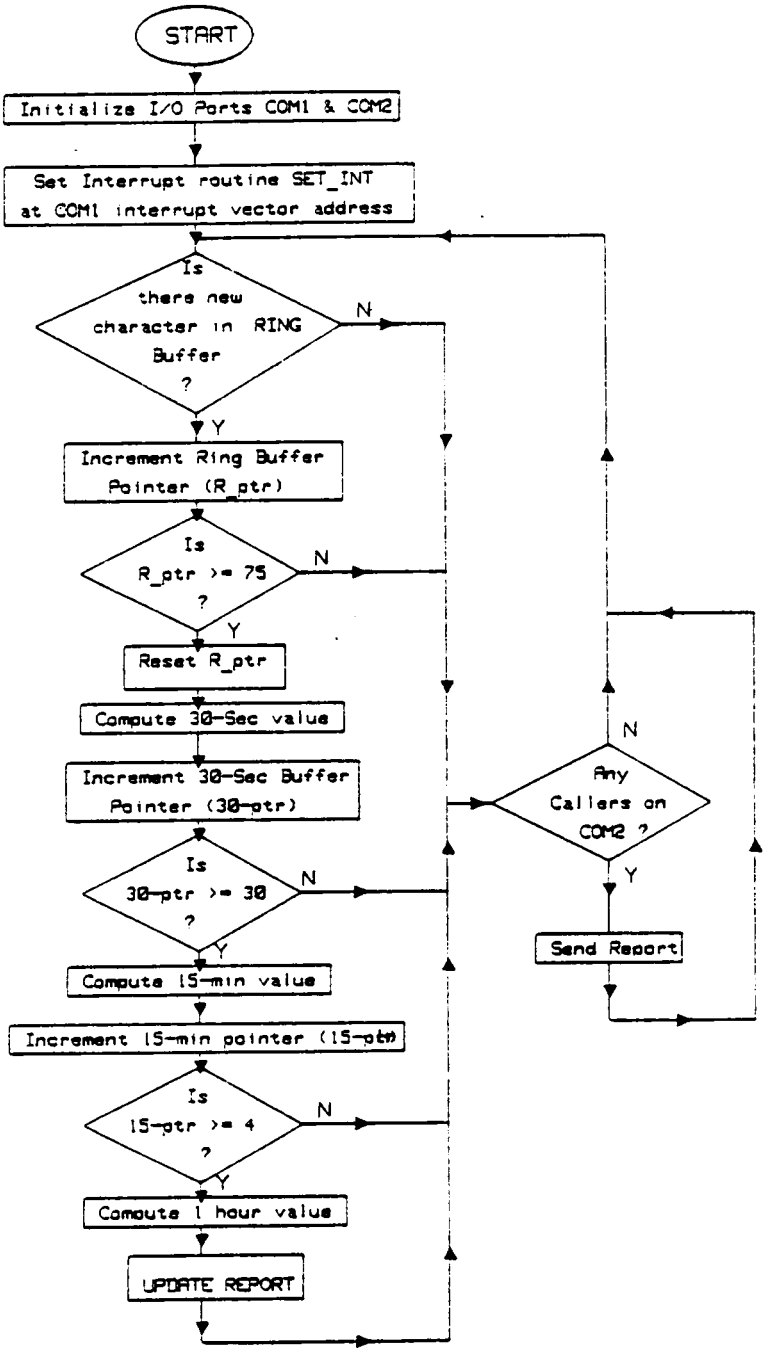

Figure 9. Data Acquisition Flowchart.

4.3.1.1 Polling technique in COM_INT.

The line-status register (LSR) is polled every time there an interrupt COM1. The status returned by the LSR is checked for the setting of the zero-th bit. If no data is available, the **DTR** is asserted and an End-Of-Interrupt(**EOI**) is sent to the 8259 communication board COM1 reg.0 to enable interrupts.

4.3.1.2 Filtering Technique in COM_INT.

Noise or interference in transmission distorts data characters and if these are not detected and suppressed, they result in erroneous data. As mentioned in chapter 3, leased lines guarantee error free data upto a certain extent, but that is not enough. Hence to ensure complete error free data a filtering technique was developed in this project and has been outlined below.

Since the first character of a valid data value is a '+' sign, the software filter checks for a '+' sign for a valid start of an eight-digit value. If this '+' sign is available, the next seven digits are checked to see if they are a digit between '0' and '9' or a '.' or a space character (Ascii value HEX 20) or a <CR> (Ascii value HEX 0D).Any character other than the ones mentioned above sets a flag indicating bad data value and the load data in the ring buffer is held at the

previous value. If all eight digits are valid data values, they are placed in HR_BUF and the buffer pointer is incremented. If there is any indication of bad character, the load data updates are held at the previous value and the filter starts looking for a '+' sign for a fresh start of valid data. (For software flowcharts refer to "Appendix A. Data Acquisition software." on page 92).

The ring buffer is externally declared in the main program and hence is available to the main program as well as to the COM_INT routine. The COM_INT fills the buffer HR_BUF at a different rate compared to the rate at which the main program reads it. Since data comes in at the rate of five load data values in two seconds, this is the rate at which the buffer is filled and the main program reads at a faster rate. To avoid overlaps, 30 seconds of data is collected by taking 75 values of the HR_BUF and averaging it (since 30 x 2 1/2 = 75). This 75-value window moves forward without overlapping and at the end of 30 75-value windows, the ring buffer pointer is reset to zero. On the 3B2, the 30 seconds was fixed by a real time clock and this was strictly a 30-second window irrespective of the number of data values arrived.

4.3.2 Subroutine INIT COM.

This routine programs the 8250 registers that control the I/O ports COM1 and COM2. Initially all the registers are

read to clear them. Then the DLAB bit (the 7th bit) of the Line Control Register is cleared to enable read and write of the transmitter buffers. The Interrupt Enable Register is set to enable recieve data and recieve line status interrupts interrupts. The Modem Control Registers (MCR) are set to assert the DTR and RTS output signals of the 8250. The MCR for COM1 (address 3FC) has it's bit 2 set to directly connect the serial connect the serial port to the RS 232-C interface. The MCR for COM2 (address 2FC) has bit 2 reset to allow the port to be controlled by the internal Metrabyte modem. This subroutine is called in the main program after initialization of the ports using COM_RST.

4.3.3 Subroutine PUTRS.

This routine polls the Line Status Register (LSR) of the port involved for transmitting character out. The bit 1 of the LSR when set indicates that transmit buffer is empty. If set, the character to be transmitted is sent to the transmitter buffer. The input requirements of this routine is the character to be transmitted.

Declarations:

```
int putrs(ch);

char ch;
```

4.3.4 Subroutine GET RS.

This routine regulates the buffer pointer of a buffer which is incremented every time a data is available on the com port whether it is command responses or load data. This buffer is different from the HR_BUF buffer as the latter just collects load data. For details on this routine refer to program flowchart listing "Appendix A. Data Acquisition software." on page 92

4.3.5 Subroutine POLLING.

This is a major subroutine that controls the transmission of the report to callers on COM2. It checks for a one character userid of the callers before transmission begins. This is done by continuously polling the LSR of the COM2 port for incoming character. If the userid is verified, the transmission begins. Since it is an uniprocessing system a character is transmitted each time the major loop of the main program is traversed. Such a method does not interrupt the data processing in the main program in order to transmit report and therefore can be compared to a time-sharing process up to a certain extent. The details of the report sent is discussed in "Results of Data Acquisition." on page 79

4.3.6 The Main program.

This is the main body of the Data Acquisition System Software and it is a uniprocessing system.This means that a single major loop handles all data processing and data transfer tasks.

4.3.6.1 Data processing.

Previous to entering the main loop, data files are read to arrange the file pointers for future update. The console is updated with the last 24 hours data and the current monthly peak, all updated from existing data files. Once the program starts executing, these data files are updated with current data.

In the lease line data collection software, there are no modem commands, and so the data starts to arrive immediately after connecting the RS 232-C interface to the link. In the dial-up line commands are sent to the Penril modem on COM1 to dial or hang-up or redial as required. Responses from the modem are collected from the COM1 port as incoming data but differentiated from load data by placing them in a separate buffer COMMAND. It has been assumed that once load data starts to arrive the modem is no longer in command mode and hence incoming data is only load data.

When load data arrives, the main program starts checking for 75-data values in the HR_BUF. Once 75 values have arrived, they are averaged for a 30-second value. This averaging also checks for any erroneous variations in the data. If the difference between two consecutive load values is greater than 1500 MW, the sudden change is suppressed by neglecting the change and holding the load value at the previous valid load value. This takes care of resets or sudden surges in the data. For example, if the filtering technique in the COM_INT routine passed on a load value that consists of valid characters but is about 1600MW higher than the previous load value. Such a variation is obviously an error that did not get filtered. The averaging of 30 second value in the main program suppresses such variations by holding the load value at the previous one. Thus if three load values are : 4000MW,5600MW,4004MW, the average of these three is 4001MW by the above method.

After a 30 second value is computed, a check is made for a 15-minute data value. This simply achieved by counting 30 30-second data values and averaging them. The 15-minute data values are stored to form historical database for the predictor program. Similarly, hourly values are computed, by taking average of 4 15-minute values, and are stored in active buffers and data files. The storing of data in files is necessary so that data is not lost in case of a sudden power failure. 24 hour data values are tabulated along with four

15-minute values of the latest hour, for transmission to callers. The details of the report format is discussed in "Results of Data Acquisition." on page 79.

4.3.6.2 Data transfer.

This deals with the transfer of load data to COM2 using the subroutine POLLING. The monthly peak and 24 hour load data is sent as report. The transmission of characters is handled by the POLLING routine. The main program organizes the data into strings that are passed to POLLING for transmission. The transmission is ended by commanding the Metrabyte modem connected to the COM2 port to hang-up.

Three macros of the C-86 compiler have been used for I/O port handling namely:

1. COM_RST;

   This sets up parameters at the specified for communications.

2. INTRINIT;

   This sets up the interrupt routine at the interrupt vector specified and allocates a stack size too.

3. INTRREST;

This restores the routine specified to the same interrupt vector set up by the INTRINIT.

4.4 CONCLUSIONS.

A uniprocessing system is not very efficient when complicated situations arise. An attempt to fit a multitude of tasks in one loop slows the system down and in a real time application that is costly loss. Yet, if a multitasking system is not available, the above problem can be solved by introducing multi-processing systems on the PC.

4.4.1.1 P.C as mainframe workstation.

The use of P.C as mainframe workstations is still largely in it's infancy. In this particular role, the P.C is considered permanently connected to the mainframe host, merely complementing the host's processing and storage capabilities with it's own so the user sees the whole system as a smoothly integrated environment. The critical factor in this type of environment is that the P.C must be able to handle mainframe communications and add other functions as well.

The advent of co-processor systems for assisting in communications support has made it easier to process the host data locally, but the present single-tasking operating sys-

tems need careful handling to achieve reasonable performance. The new multitasking operating systems like the concurrent CP/M-86 and the IBM's newly announced TOPVIEW to run with it's PC-DOS 3.1 will provide a much more sophisticated environment on which to run Workstation software and also provide a significant improvement in PC workstation capabilities.

5.0 DATA ACQUISITION SUBSYSTEM ON THE 3B2/300 SUPERMICRO.

A real time application system usually consists of a number of co-operating tasks. When the tasks become too complicated to be handled by a single foreground - background task and interrupts, multitasking is a useful solution. The Data Acquisition software for load and weather data is of multitasking type, consisting of a number of multiple tasks held together by the UNIX operating system.

The multitasking characteristic of UNIX OS provides the 3B2 computer with the capability of executing more than one program simultaneously [11]. Hence Data Acquisition of load data, weather data collection and the load forecast program can all be executed at the same time as background and foreground processes without interrupting each other.

The UNIX system supports time sharing which is a kind of multitasking system designed to support multiple, interactive users on terminals [11]. This is very useful as the users can share the archived load, weather data and forecasted result simultaneously independant of any interruption except for power failure.

I/O communication on the 3B2 is of utmost importance in this project as it is the basic task of the Data Acquisition Subsystem. The next section explains the I/O capability offered by UNIX on 3B2/300 supermicro.

5.1 I/O HANDLING ON THE 3B2 SUPERMICRO.

The hardware characteristics of the 3b2 have been discussed in "Central Computer." on page 11.The I/O ports can be accessed directly using the Basic Networking Utilities offered by AT&T (refer to "Basic Networking Utilities (BNU)." on page 56) [6] or by system software (refer to "UNIX System calls for I/O handling on 3B2." on page 63). In this project the weather data collection was achieved using the BNU. The archiving of weather data was done on Fortran 77 language. The I/O handling of load data acquisition, as well as archiving of the data acquired, was written in C-language using the various system calls offered by UNIX [12]. (listed in "UNIX System calls for I/O handling on 3B2." on page 63). This section highlights the important characteristics of BNU which were used in weather collection. It also describes the different system calls in UNIX, which were used in load data acquisition software.

5.1.1 Basic Networking Utilities (BNU).

The BNU is composed of software programs, daemons (background routines) and a supporting data base. This utility allows the host computer to interactively communicate with the UNIX system machines and non-UNIX system machines [12].

5.1.1.1 BNU Hardware requirements.

Before the 3B2 can communicate with a remote system, a communication link must be established to the remote machine. The communication link used in this project to collect weather data from a weather forecasting service using BNU, is a switched line with telephone modem that has an ACU (Automatic Call Unit) [1]. The ACU dials the specified telephone number upon request from the BNU.

5.1.1.2 BNU Software.

There are several directories that contain the BNU software and support files of Basic Networking. Two of the directories in the UNIX OS that are used most often by the BNU software are listed below:

1. **/USR/BIN.**

   This directory is used by the UNIX OS to store executable programs and is used by the Basic Networking System for the same purpose.

2. **/USR/LIB/UUCP.**

   This directory contains files of the supporting database that the BNU uses to dial out to the specified

modem using a certain specified protocol. (refer to next section for details of specified protocol).

Supporting Database Files.: Several of the Basic Networking programs require information contained in support files. These files are located in the /USR/LIB/UUCP directory. The following two files are important in establishing communication using the 'CU' :

- **Devices.**

  This file contains information about the location and line speed of the ACU, the direct and other possible devices linked to the I/O ports of the computer.

  A typical entry for the /USR/LIB/UUCP/Devices file:

  ACU TTY*-1200 Penril where the TTY stands for Terminal Type and the * is replaced by the corresponding port number. This entry specifies speed and location of a Penril modem.

- **Dialers.**

  This file contains character strings required to negotiate with Network Devices in the establishment of connection to remote computers. This initial handshaking is usually a sequence of ASCII strings that are transmitted and expected. The entry in a typical Dialers file

for a Penril are explained below. The BNU is compatible with quite a few popular modems such as Penril, Ventel, Hayes, Develcon. The modem used in this project is a Penril. The Penril entry in the Dialers file is as follows:

```
Penril=W-P"" \d>s\p9\c)-W\p\r\ds\p9\c-)y\c : \E\TP >
9\c OK
```

The penril entry shown above is executed as follows.

First, the phone number arguments is translated, replacing any "=" with a (pause). The handshake given by the remainder of the line works as follows :

""	Wait for nothing.
\d	Delay for two seconds.
>	Wait for a ">".
s\p9\c	Send a 's',pause for 1/2 second, send a 9 and no terminating new-line.
)-W\p\r\ds\p9\c-)	Wait for a ')'. If not received process with the string between the characters '-' as follows. Send a 'W',pause, a carriage

return,delay,'s','9',no new-line and wait for the ')'.

y\c	Send a 'y'.
:	Wait for a ':'
\E\TP	Enable Echo checking. Then send phone number followed by a pause.
>	Wait for a '>'
9\c	Send a '9' without a new-line.
OK	Waiting for a 'OK' string.

A very important feature in the BNU is the 'CU' command. The 'CU' user program connects the 3B2 to a remote machine and allows transfer of files or execution of files on either machine without dropping the connecting link. It can use direct links, the telephone network, or the LAN to establish connection. Appendix B lists the options on the 'CU' command for a quick reference. The reader may refer to the BNU guide [6] for further details.

5.1.2 Weather Data Acquisition.

The Weather Data, as mentioned in earlier chapters, was collected from a weather forecasting service at specific times of the day. The 'CU' command of the BNU was utilized for dialing out using a Penril modem as the peripheral.

The Weather Data Collection required an extended ASCII string for the handshaking protocol in the Penril entry of the Dialers file. Since the data had to be collected as a background process, and at specific times if the day, it was necessary to have the least possible user interaction. Secondly, for collection of data, a specific logon procedure to be followed by every user on it, and this had to be sent by the 3B2 before actual data was made available. As a result the Dialers file entry for the Penril Modem was modified to include the logon requirements.

The necessity of being user independant required the use of the **'CRON'** daemon, offered by the UNIX OS, to automatically dial out at specified times. The **'CRON'** daemon handles the sequence of executable files listed in the /USR/SPOOL/CRON/ CRONTABS/ROOT file [12]. When the 3B2 is in multiuser mode, the CRON process checks the above file for changes once every minute. The line entry format for a CRONTAB file is as follows:

MIN HOUR DAY MONTH YEAR COMMAND.

The Command section consists of the names of commands which may be UNIX shell commands, BNU commands, or user-created executable files. In this project, the command section of the root crontab file were:

- 'CU' command of the BNU for dialing out to weather service.

- Since the 'CU' command does not hang up without user interaction, the DTR of the line was made to drop by using a user-created program called HANG-UP (Details listed in "HANG_UP.C" on page 73) In the root crontab file, this HANG-UP program was executed after allowing ten minutes to pass from the start of the 'CU'. This was done under the assumption that data collection would be completed in a time span of ten minutes.

- Executable Fortran programs for load data archiving and creating historical database for the load forecast program.

The Weather data consists of actual data for the current day and the forecasted data for the next day. This data is initially stored in a temporary file. The type of archiving attempted on the raw data depended on the time it was collected. Essentially, two types of files were formed. One was

for the users on the system to view, and the other formed the historical database required as input to the load forecast code. Chapter 6 lists the format of the 'WEATHER REPORT' which is the archived file for users to view.

5.1.3 UNIX System calls for I/O handling on 3B2.

As mentioned in "Central Computer." on page 11the 3B2 supports I/O hardware circuitry that can be extended to 18 serial ports and 4 parallel ports. The ports can be acessed by the users as special terminal files. The **READ and WRITE** system calls of the UNIX OS enable the user to exchange information to and from the I/O ports, assuming proper initialization of parameters have been completed prior to data transfer [12]. This section, at first, explains the software techniques used to access the I/O ports and any devices connected to them. Then an attempt has been made to explain the various subroutines that link together to form the Data Acquisition Software for Load collection.

5.1.3.1 Software for Terminal Interface on 3B2.

All of the asynchronous communication ports use the same general interface, no matter what hardware is involved.

When a terminal file is opened, it normally causes the processes to wait until the connection is established. UNIX

supports a number of application programs, accessible to any user [12]. These system calls assist in opening the desired port as a terminal file, initializing the communication parameters on it and allowing transfer of data between host computer and remote system. Four of the application programs are listed below :

- **OPEN.**

  - Declaration : INT OPEN(path,oflag [,mode]);

  - Files included: #include <fcntl.h>

  - Description: 'PATH' points to a path name naming a file. All serial ports have the corresponding terminal files in the **/DEV/** directory. OPEN opens a file descriptor for the named file and returns the file descriptor. 'oflag' is formed by or-ing a few status flags. Listed below are the two flags used.

    — **O_RDWR.** File opened for read and write.

    — **O_NDELAY.** When opening a file associated with a communication line, if O_NDELAY is set, the OPEN return without waiting for carrier.

- Return Value: Upon successful completion, a non-negative integer, namely the file descriptor is returned. Otherwise a value of -1 is returned.

- **IOCTL.** [11].

  - Declaration: INT IOCTL(FILDES, COMMAND, POINTER); STRUCT TERMIO *POINTER;

  - Description: This code initializes the communication parameters of the port opened by the OPEN system call.It's input requirements are:

    1. FILDES. This is file descriptor returned after opening the port as a terminal file using OPEN.

    2. COMMAND. The commands used are:

       — TCGETA.
       — TCSETA.
       — TCSETAW.
       — TCSETAF.

       In this project, the command used is TCSETAW. This command waits for the output to drain before setting the new parameters.

3. POINTER is a pointer to a structure TERMIO, defined in <TERMIO.H>.

The structure TERMIO defines control flags that set the modes, communications parameters, and control characters of the I/O port.

```
#define NCC    8
STRUCT termio {
Unsigned short c_iflag;     /* input modes     */
Unsigned short c_oflag;     /* output modes    */
Unsigned short c_cflag;     /* control modes   */
Unsigned short c_lflag;     /* local modes     */
char           c_line ;     /* line discipline*/
unsigned char  c_cc[NCC];   /* control chars   */
};
```

The following settings were used for Load Collection.

1. The c_iflag was set to 'ICRNL' which mapped a <cr> (carriage return) to a <nl> (newline) on input.

2. The c_cflag describes the hardware control of the terminal. This flag was set to:

a. 1200 baud.

b. 7 data bits.

c. Dial-up type connection line.

d. enable character receive.

e. Drop DTR when parent process terminates and returns to UNIX shell.

After setting the above flags in the structure TERMIO, a system call to IOCTL with a file descriptor from OPEN sets the parameters for the specified port.

- Return Value. If error has occured, a value of -1 is returned, else control is passed to the next executable statement of the code.

• READ [12]

- Declaration. INT READ(fildes, buf, nbyte);

- Description:

Read attempts to read nbyte bytes from file associated with the file descriptor **fildes**, returned from the OPEN system call.

When attempting to read a file associated with a TTY port that has no data currently available, if the O_NDELAY flag in the OPEN call has been set, the READ will return a zero. This is useful when polling a port for available data.

- Return value. Upon successful completion, a non-negative integer is returned indicating number of bytes actually read. In case of error, a value of -1 is returned.

• WRITE [12].

- Declaration. INT WRITE(fildes, buf, nbyte);

- Description:

Write attempts to write nbyte bytes from file associated with the file descriptor **fildes**, returned from the OPEN system call.

When attempting to write to a file associated with a TTY port that has has no more space for new data, if the O_NDELAY flag in the OPEN call has been set, the WRITE will return a zero.

- Return value. Upon successful completion, a non-negative integer is returned indicating number of bytes actually written. In case of error, a value of -1 is returned.

5.2 SYSTEM SOFTWARE FOR LOAD DATA ACQUISITION.

Figure 10 on page 70 is a simplified flowchart of the Load Data Acquisition software. The **PENRIL.C** handles the I/O protocol of the 3B2. It utilizes the OPEN, IOCTL, READ and WRITE system calls explained in "UNIX System calls for I/O handling on 3B2." on page 63. Before attempting to explain the logical flow of the various subroutines, two more UNIX system calls have been discussed below. These are used for communicating between processes which are running simultaneously.

- **FORK.** [15]

  - Declaration : INT FORK;

  - Description : This causes creation of a new process.The new process (child process) is a exact copy of the calling process (parent process).

    The child process has a unique process ID and it's own copy of the parent's file descriptors. Hence

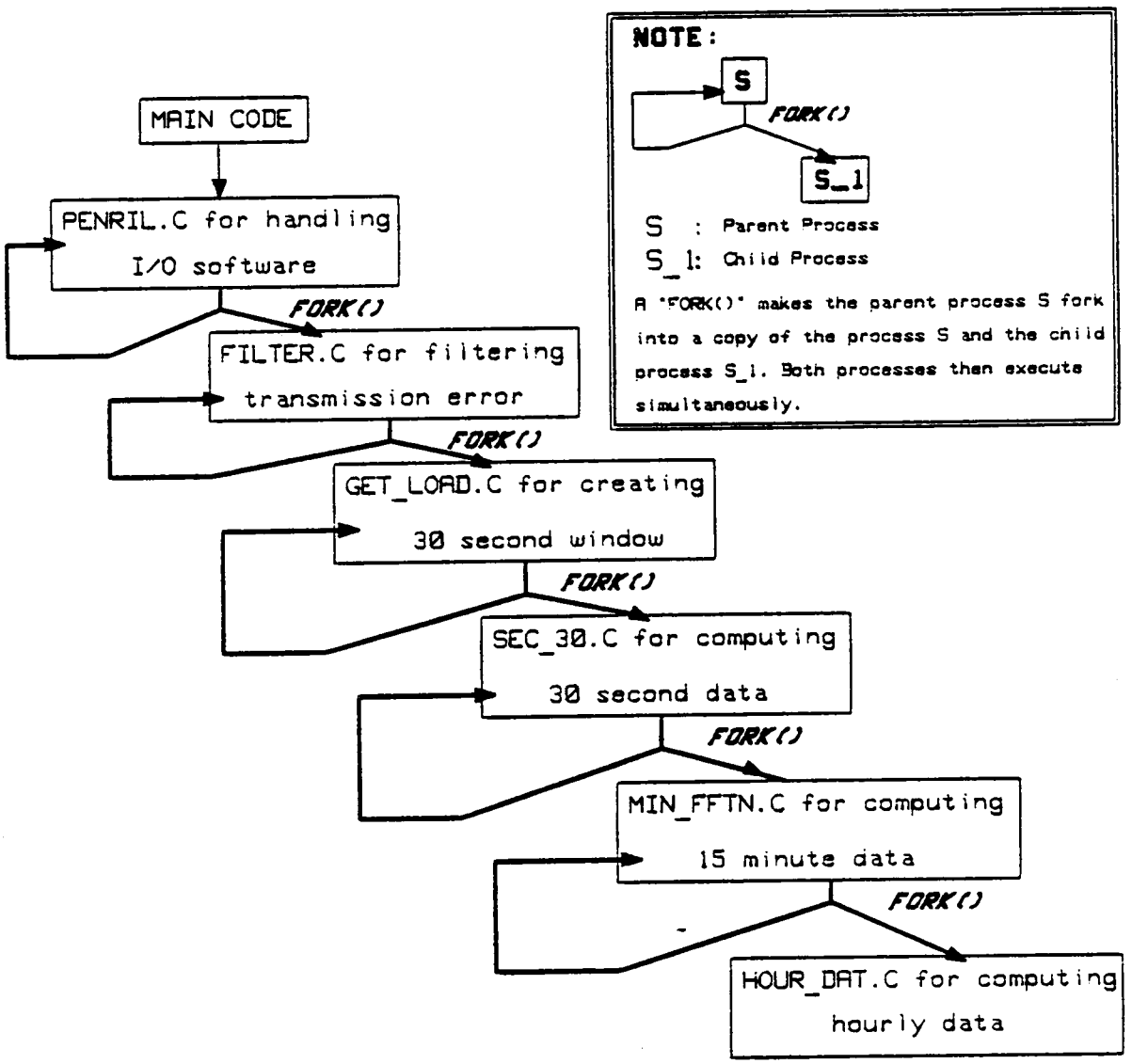

Figure 10. Load Data Acquisition Flowchart.

Data Acquisition subsystem on the 3B2/300 supermicro. 70

a Pipe created in parent process connects the two forked processes thereby enabling transfer of data.

The syntax of forking in a parent process is as following:

```
IF (( FORK()) == 0) CHILD_PROCESS();
```

- Return Value. Upon successful completion, **FORK** returns a value of 0 to the child process, and the process ID of the child to the parent. In case of error, a value of -1 is returned to the parent process and no child process is created.

- **PIPE.** [12]

Declaration : INT PIPE(fildes);

Description :

This creates a interprocess channel called a pipe. It returns two file descriptors, fildes(0) and fildes(1). Fildes(0) is opened for reading and fildes(1) is opened for writing.

Writes upto 5120 bytes of data are buffered by the pipe before the writing is blocked.

When a parent process forks into a child process, a PIPE system call helps in exchanging data be-

tween two processes without interrupting the execution of either process.

Return Value. Upon successful completion, value of 0 is returned. In case of error, a value of -1 is returned. A PIPE call fails if 19 or more file descriptors are currently open.

5.2.1 Software Development.

All data shared by parent and child processes are through pipes created by the parent process using the PIPE system call. Since the O_NDELAY flag has been set while opening the TTY11 file, all reads and writes from interprocess pipes has been done using polling technique. The child process waits for data to be available on the channel before processing it.

This section discusses the subroutines mentioned in the Figure 10 on page 70 The software flowchart are included in Appendix A. The flowcharts are listed along with helpful documentation.

5.2.1.1 PENRIL.C.

This subroutine mainly deals with the I/O handling of the device port TTY11 which is connected to the telephone

network through the PENRIL modem. The system calls mentioned in "UNIX System calls for I/O handling on 3B2." on page 63 are used to open the TTY11 as a terminal file with the path name as **/DEV/TTY11.** and set the proper control flags of the **TERMIO structure** . After completion of handshaking protocol with the Penril modem, when the modem goes in originate mode, the software starts checking for a '+' character to signal the start of valid load data. When a '+' is received, the PENRIL.C forks into a child process FILTER.C that filters the stream of raw characters for transmission errors and assembles them to form a valid load data of the form : "+***.***" where the * is replaced by any digit from 0 to 9.

5.2.1.2 HANG_UP.C

This program is used to hang up communicating links by dropping the DTR to a low level. It accesses the UNIX system call IOCTL and sets the c_cflag in the TERMIO structure to a zero Baud. This setting controls the DTR of the link and drops it if set to a zero value. Refer to Appendix A for the program listing.

5.2.1.3 FILTER.C.

The filtering technique is the same as that employed on the IBM PC-XT for data acquisition. "Filtering Technique in

COM_INT." on page 46 explains in detail the design of the technique and hence another discussion on it has been omitted in this section. The **FILTER.C** forks into a child process GET_LOAD.C that accepts the filtered data and arranges them into the ring buffer.

5.2.1.4 GET_LOAD.C

This subroutine creates the 30 second window on the ring buffer. After the data has arrived, the ring buffer gets loaded 2 & 1/2 times every second with valid data and a pointer keeps track of the address of the data in it. Setting of a 30-second window is done in phase with a real time clock, and hence the pointer of the ring plays a very important role. A slight miscalculation of the pointer will distort the final 30-second value that is computed after averaging the data values that exist within the particular 30-second window. This process forks to a child process SEC_30.C that collects the averaged 30-second value from GET_LOAD.C.

5.2.1.5 Routines that archive the processed load data

The codes SEC_30.C, MIN_FFTN.C and HOUR_DAT.C compute 30-second load data, 15-minute load data and hourly load data respectively.

SEC_30.C collects data every 30 seconds from a interprocess pipe from GET_LOAD.C and stores it in a 24-line data file which scrolls up with the latest load update. It also averages 30 30-second data values to compute the 15-minute load value and passes it to it's child process MIN_FFTN.C.

MIN_FFTN.C collects load data every 15 minutes from a interprocess pipe from SEC_30.C and stores it in a 24-line data file which scrolls up with the latest 15 minute load update. It also averages 4 15-minute data values to compute hourly load value passes it to it's child process HOUR_DAT.C.

HOUR_DAT.C collects load data on an hourly interval and stores them in two files. One of them is a 24-line file which scrolls up with the latest hourly load update. The second created for database purposes. All 24 hour load values for a day are stored in a file that appends it current day load value to the previous day's file.

"Results of Data Acquisition." on page 79 lists the format of the reports prepared by the above mentioned routines.

5.2.1.6 GETDATE.C

This subroutine is the heart of the software as all computations are time dependant and they require the system real time clock to time each process.The C-compiler on the 3B2 supports a few time functions which are used for access-

ing and reformatting the systems idea of the current date and time. These functions are located and loaded automatically when a C-language program is compiled.

The GETDATE.C code utilizes these time functions to access the system clock. To do so, a header file associated with these functions is required to be included at the top of the program that is being compiled. The first line of the code is therefore:

```
#include <time.h>
```

The time function in GETDATE.C is

- Declaration `#include <time.h>`
- `long *clock;`
- `struct tm *localtime(clock);`
- Description
- The **LOCALTIME** returns a pointer to the structure of type tm [3]. The structure tm in the header file time.h is as listed:

```
Struct tm {
int tm_sec;
int tm_min;
int tm_hour;    /* hour of day ( 0 - 24 ) */
int tm_mday;    /* day of month ( 1 - 31) */
int tm_year;    /* month of year ( 0 - 11) */
```

```
int tm_year;      /* last two digits of current year
*/
int tm_wday;     /* day of week ( Sunday = 0 ) */
int tm_yday;     /* day of year ( 0 - 365 )  */
int tm_isdst;    /* nonzero if DST is in effect */
};
```

The GETDATE.C calls the time function LOCALTIME which returns to GETDATE.C the pointer to the tm struct [13]. The current time and date is then got by accessing the elements of the structure listed above, using the pointer returned by the LOCALTIME.

5.2.1.7 Conclusion.

The 3B2 C-compiler supports a wide variety of miscellaneous functions that assist in making the code efficient and compact. The I/O handling capability of the UNIX system has proved to be very useful in load data collection. For weather data collection, the BNU was utilized as the weather service required a LOGON procedure with userid and password which required special considerations on the Penril. The 'CU' has built in self diagnostic capability which was required when using the Penril for sending text to the remote system before going into Originate Mode. The system software written for load data does not have as sophisticated diagnostic methods

as the 'CU'. Therefore the 'CU' was preferred for weather data collection. In future, the data communications software written in this project will be modified to include logon capabilities and other timing considerations.

6.0 RESULTS OF DATA ACQUISITION.

The data that was collected and filtered for transmission errors, had to be archived and stored in a tabulated form for other users to view. On the IBM PC/XT, the only data available was the archived load data. On the AT&T 3B2/300 supermicro, the weather as well as the load data were made available to user.

This chapter discusses the format of the tabulated data prepared for the viewers on the IBM PC/XT and the 3B2.

6.1 THE IBM PC-XT REPORT.

Figure 11 on page 80 gives an example of the report that was available on the COM2 port of the IBM PC. Any off-site computer can call into this port and view the report after clearing proper security code. The first line is the monthly peak updated every 24 hours at 00:00:00. On the first day of the month at 00:00:00 hour, the previous monthly peak is substituted by the load at that hour, thus starting a new monthly peak. After 24 hours this peak is updated at 00:00:00 hour if required. The second line consists of two load data values:

- Hourly load data

```
                    9666 Indicated peak: 17:00:00  9/09/1985
4394 22:02:46  9/15/1985       5620 11:15:32
4354 23:02:48  9/15/1985       5590 10:33:05
3921  0:02:50  9/16/1985       5606 10:48:05
3634  1:02:51  9/16/1985       5615 11:03:06
3496  2:02:53  9/16/1985
3463  3:02:55  9/16/1985
3442  4:02:56  9/16/1985
3549  5:02:58  9/16/1985  Projection available.
3898  6:02:59  9/16/1985
4808  7:03:01  9/16/1985
5341  8:03:02  9/16/1985
5438  9:03:04  9/16/1985
5550 10:03:05  9/16/1985
5994 11:03:06  9/16/1985
4658 12:02:32  9/15/1985
4662 13:02:33  9/15/1985
4620 14:02:35  9/15/1985
4553 15:02:36  9/15/1985
4503 16:02:38  9/15/1985
4560 17:02:39  9/15/1985
4644 18:02:41  9/15/1985
4684 19:02:42  9/15/1985
4942 20:02:44  9/15/1985
5049 21:02:45  9/15/1985
```

Figure 11. The report for Data Acquisition on the IBM PC.

- Fifteen minute load data.

The latest hour is shown as an hour value as well as four fifteen minute values. The report is of 24 lines, showing load data for the last 24 hours including 4 fifteen minute data values of the most recent hour.

The I/O handshaking protocol of COM2 port on the IBM PC/XT has been handled by the Metrabyte Modem card. The Figure 7 on page 41 shows the switch settings on the Metrabyte modem that changes the serial port connected to the modem from COM1 to COM2 and vice versa. These settings were changed on one of the modem cards to make the serial port COM2. Sending of the data over the serial port COM2 was done by polling. The Data Communication software for Data Acquisition on the IBM PC/XT programmed the modem on COM2 to disconnect the link after transmission of twenty four lines of data. This automatic disconnection was installed due to unavailability of multiuser characteristic on the IBM PC/XT. Also, this helped in clearing up the COM2 port for other callers to view the report on he same port.

6.2 THE 3B2 REPORT.

Due to multiuser nature of the UNIX OS, the tabulated form of the data is available to any viewer who is logged on to the system. The \BIN\ directory in the UNIX OS contains

all public commands. Hence any executable file can be used as a command globally if it exists in the \BIN\ directory. The following commands were created for load and weather reports on the 3B2:

- " **Load** " for viewing 30-second, 15-minute and hourly load data file.

- " **Weather** " for viewing latest update of weather data as reported by a commercial weather reporting organization.

- " **Forecast** " for viewing the prediction results in tabulation form.

6.2.1 Load Report.

This report is stored as three 24 line data files in the /ETC/ directory. It scrolls upward with the latest updated value. Three types of load data files form the LOAD report. They are:

1. 30-Second data file.
2. 15-Minute data file.
3. Hourly data file.

```
5000 23:00:00
5033 23:00:30
5037 23:01:00
5043 23:01:30
5044 23:02:00
5046 23:02:30
5049 23:03:00    This is the 30-Second Load Data file
5050 23:03:30
5050 23:04:00  The latest load update is the last line
5050 23:04:30
5053 23:05:00     The file scrolls up every 30-seconds
5052 23:05:30
5050 23:06:00
5053 23:06:30
5049 23:07:00
5052 23:07:30
5051 23:08:00
5050 23:08:30
5054 23:09:00
5054 23:09:30
5054 23:10:00
5054 23:10:30
5054 23:11:00
5054 23:11:30          Today's date  8/18/1985
```

Figure 12. The 30-second data report available on 3B2.

```
5925 18:00:00
5909 18:15:00
5905 18:30:00
5885 18:45:00
5869 19:00:00
5883 19:15:00
5870 19:30:00    This is the 15-Minute Load Data file
5973 19:45:00
5901 20:00:00  The latest load update is the last line
6001 20:15:00
6010 20:30:00     The file scrolls up every 15 minutes
5979 20:45:00
5948 21:00:00
5918 21:15:00
5871 21:30:00
5822 21:45:00
5771 22:00:00
5714 22:15:00
5618 22:30:00
5492 22:45:00
5378 23:00:00
5296 23:15:00
5171 23:30:00
5054 23:45:00      Today's date 8/18/1985
```

Figure 13. The 15-minute data report available on 3B2.

```
5460 00:00:00
4999 01:00:00
4661 02:00:00
4435 03:00:00
4319 04:00:00
4242 05:00:00
4221 06:00:00    This is the Hourly Load Data file
4294 07:00:00
4397 08:00:00  The latest load update is the last line
4809 09:00:00
5300 10:00:00     The file scrolls up every hour
5684 11:00:00
5830 12:00:00
5957 13:00:00
5957 14:00:00
5866 15:00:00
5797 16:00:00
5801 17:00:00
5886 18:00:00
5892 19:00:00
5882 20:00:00
5984 21:00:00
5845 22:00:00
5550 23:00:00     Today's date 8/18/1985
```

Figure 14. The Hourly data report available on 3B2.

Each of these files have 24 entries, with the latest update on the 24th line. Since these files exist in the /BIN/ directory, they are accessible to multiple users simultaneously. Hence, any user can login in to the 3B2 and view the load files using the command 'LOAD'.

The load command offers the following options:

Enter	View
30s	30 second load data file
15m	15 minute load data file
24h	hourly load data file
q	Quit

Thank You

The above command is completely menu driven, and is a shell program [11]. This command pages through the data file requested by the user and returns to the menu after each paging. The user can quit the command any time by entering 'q' as a input. Letter 'm' is entered to go back to the menu after viewing a report. This allows the user to view more than one report.

6.2.2 The Weather Report.

The weather data is stored as a 48 line data file, consisting of 24 lines of weather data of the current day and 24 lines of predicted data for the following day. This file exists in the /ETC/ and is updated frequently every day. Each update is rewritten over the previous data, thus maintaining the file length of 48 lines. This file is paged through using the command 'WEATHER'. The weather data is a collection of data for three weather stations, namely, Washington DC, Norfolk and Richmond.

The Figure 15 on page 88 shows an example of the weather file that contains archived actual data and the forecasted data.

6.3 CONCLUSIONS.

The multitasking characteristics of the UNIX OS and the 3B2 enables one to accomplish a number of tasks, all at the same time, on the same system, thereby assisting the user to stay in phase with the real time clock. Therefore, in this project, even if the communicating link is slowed down due to heavy on-line traffic, the data acquisition software maintains a 30-second window on the data coming in, independant of the data rate.

MONDAY 23-SEP-85 5:19 AM

Time YYMMDDHH	WASHINGTON TMP	DEW	Wind DIR	MPH	CLD	RICHMOND TMP	DEW	Wind DIR	MPH	CLD	NORFOLK TMP	DEW	Wind DIR	MPH	CLD
85 923 0	65	65	45	6	8	64	55	0	8	8	71	69	45	13	8
85 923 1	65	65	0	7	8	64	63	315	10	8	72	69	45	9	8
85 923 2	66	65	0	8	8	64	63	0	9	8	70	67	90	8	8
85 923 3	66	66	45	7	8	64	63	0	8	8	70	67	45	8	8
85 923 4	66	66	45	7	8	64	64	0	7	8	70	65	0	12	8
85 923 5	67	67	0	6	8	65	65	315	7	8	69	65	0	14	8
85 923 6	67	66	0	6	8	65	65	315	7	8	69	65	0	14	8
85 923 7	67	66	0	6	8	65	65	315	7	8	69	65	0	14	8
85 923 8	68	65	0	7	8	65	64	0	6	8	70	67	0	13	8
58 923 9	69	65	0	7	8	67	64	0	6	6	70	68	0	14	8
85 92310	70	65	45	7	6	69	65	45	7	6	71	69	0	14	8
85 92311	71	65	45	8	4	71	65	45	7	4	72	69	0	13	8
85 92312	73	65	45	9	4	74	65	45	6	4	73	69	0	13	8
85 92313	74	65	90	10	4	76	65	90	6	4	74	69	0	12	8
85 92314	75	65	90	10	4	78	65	90	7	4	75	69	0	12	6
85 92315	76	65	90	9	4	79	65	135	8	4	76	69	0	11	6
85 92316	77	65	90	8	4	80	66	135	7	4	76	70	0	9	4
85 92317	76	65	90	8	4	79	66	135	7	4	75	70	45	8	4
85 92318	74	65	90	8	4	78	65	135	6	4	75	69	45	8	4
85 92319	73	65	135	8	2	77	65	135	6	4	74	69	45	7	6
85 92320	72	65	135	8	2	75	65	135	6	4	73	68	45	8	6
85 92321	72	65	135	7	2	74	64	135	6	4	72	68	45	9	6
85 92322	71	65	135	7	2	72	64	135	6	4	71	67	90	10	6
85 92323	71	65	135	7	2	71	64	135	6	4	70	67	90	9	6
85 92324	70	66	135	6	4	70	64	135	6	4	69	66	90	8	6
85 924 1	70	66	135	6	4	69	64	135	6	4	68	66	90	7	6
85 924 2	70	66	135	6	4	68	63	135	5	4	68	66	90	7	6
85 924 3	69	66	135	5	4	67	63	135	5	4	67	65	90	8	4
85 924 4	69	66	135	5	4	66	62	135	4	2	67	65	90	8	4
85 924 5	69	66	135	5	4	66	62	135	4	2	67	65	90	8	4
85 924 6	68	66	135	5	4	65	61	135	5	4	67	65	90	7	4
85 924 7	68	66	135	5	4	65	61	135	5	4	67	65	135	7	6
85 924 8	69	66	135	6	4	67	62	180	7	4	68	66	135	9	6
58 924 9	71	66	135	7	4	70	62	180	9	4	71	66	135	10	6
85 92410	73	66	225	8	6	74	63	180	11	6	74	66	180	12	4
85 92411	75	66	225	9	6	76	63	180	11	6	76	67	180	12	4
85 92412	76	65	225	11	6	77	63	180	12	6	77	67	180	12	4
85 92413	78	65	270	12	6	79	63	225	12	6	79	68	180	12	4
85 92414	79	64	270	12	8	80	64	225	10	6	80	68	180	12	4
85 92415	80	64	270	12	8	81	64	270	9	8	80	67	225	12	4
85 92416	77	63	315	14	8	77	64	315	10	8	81	67	225	11	4
85 92417	76	62	315	14	8	75	63	315	11	8	79	67	225	11	6
85 92418	74	62	315	13	8	74	63	315	11	8	78	66	225	12	6
85 92419	73	61	315	13	8	72	62	0	12	8	76	66	270	12	6
85 92420	72	60	315	13	6	71	62	0	12	8	75	66	270	11	6
85 92421	71	60	315	12	6	70	61	0	11	8	75	65	270	11	8
85 92422	70	59	315	12	4	69	61	0	11	6	74	65	270	10	8
85 92423	69	58	315	11	4	68	60	0	10	6	72	64	270	10	8
85 92424	68	58	315	10	4	67	60	0	10	6	71	64	270	11	8

Figure 15. The weather data file for users.

Error free transmission is very essential for Data Acquisition. In applications such as load management, where accuracy of the load forecast can contribute towards a significant reduction in energy costs, it is of utmost importance to avoid any distortion in data. Multitasking property does not assist in reducing the distortion caused by transmission errors. These distortions can be avoided using leased lines for transmission, but leased lines do not guarantee complete error free data. As a result, a software filtering technique was developed in this project, which checked for errors in each character that was being transmitted. Chapter 3 describes his technique, while Appendix A highlights the flow of the technique. This software filter is independant of the host operating system or the machine used, as long as the communication parameters set in both are the same.

6.4 RECOMMENDATIONS.

The Data Acquisition Subsystem developed in this project utilizes the multitasking capabilities of the 3B2/300 supermicro and the UNIX OS for efficient data sharing between numerous processes.

Data sharing between processes, on the 3B2, has been achieved using the UNIX offered system calls **PIPE() and FORK()**. Chapter 4 explains in detail the function of the system calls PIPE() and FORK(). The data acquisition re-

quired data collection in phase with a real time clock, filtering of data, processing the data into 30 second, 15 minute and hourly value and storing them as various data files that are used for different purposes. To stay in phase with the real time clock is a very important requirement and is not easily achieved in a uniprocessing system. Hence the FORK() and PIPE() was used to share data between various processes executing simultaneously thus resulting in a multiprocessing software. This allowed data collection to continue independant of data processing thus maintaining a perfect real-time profile.

Self diagnostic capabilities are yet to be implemented in the Data Acquisition software developed on the AT&T 3B2 supermicro. The AT&T UNIX system offers terminal capabilities that enables the user to use cursor control on the console or any terminal. This is useful when trying to update the screen with newly archived data, or even for diagnostic purposes. If the Data Acquisition software fails, it is necessary to inform the users of the failure, and this is made possible by the terminal capabilities. A **BEEP** can be sounded by the computer during any type of software failure, and this BEEP system call is a function offered by the UNIX Terminal Utilities. The operator at the central computer can then restart the Acquisition after investigating the cause of the failure.

Another development to be added to the Data Acquisition Software is the automatic transfer between leased line software and the switched line software during leased line failure. If the leased line fails, a back-up switched line is required to continue with the Data Acquisition. The leased line software, on detection of failure in communication link, will send software signal to the back-up software to take over the communications. When the leased line is connected again, the back-up software terminates, handing the control back to the leased line software.

APPENDIX A. DATA ACQUISITION SOFTWARE.

The Data Acquisition was attempted on two computers, the IBM PC/XT and the AT&T 3B2/300 supermicro. The software for the IBM PC/XT consists of one main loop, and the flow chart has been included as Figure 9 on page 45. The 3B2 Data Acquisition software consists of numerous processes that execute simultaneously while sharing data with each other through Data Pipes. The main flow chart of this software has been included in Figure 10 on page 70.

This appendix lists one major flowchart of the Data Acquisition software on the IBM PC/XT. This is the COMINT program that deals with the data collection from the COM1 port. The execution of the main program listed in Figure 9 on page 45was interrupted whenever data arrived at COM1. The interrupt subroutine is the COMINT code. The routine polls for data in the COM1 port, filters it for transmission errors, stores it in the ring buffer and resets the Interrupt Enable register of the 8250 chip.

The Figure 16 on page 93 shows the flow of the COMINT subroutine. The details of the filtering technique has not been outlined in this flowchart. The reader may refer to "Data Communications for the IBM Personal computer." on page 30 for details of filtering technique.

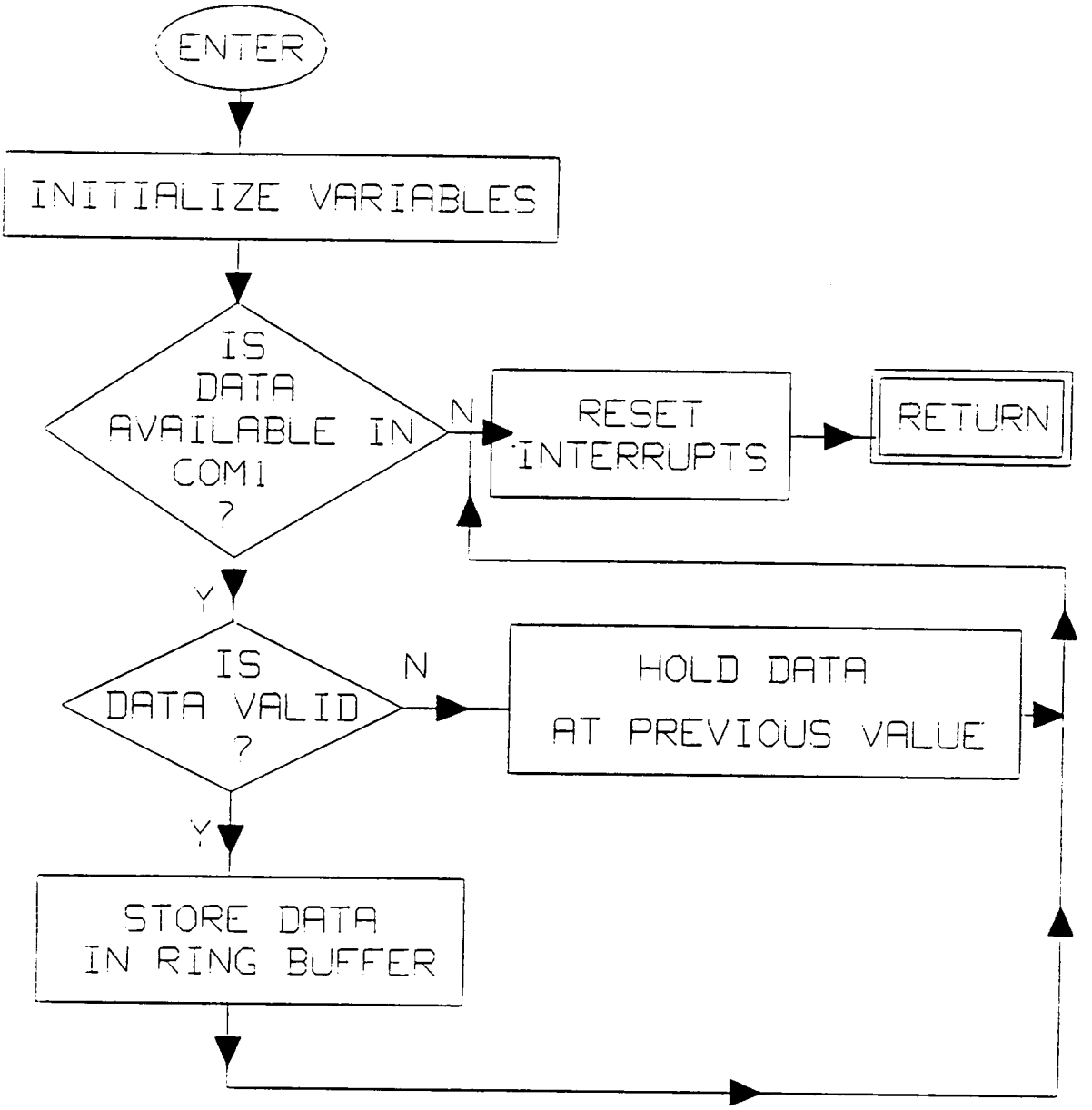

Figure 16. Flowchart for the COM1 port interrupt subroutine COMINT.

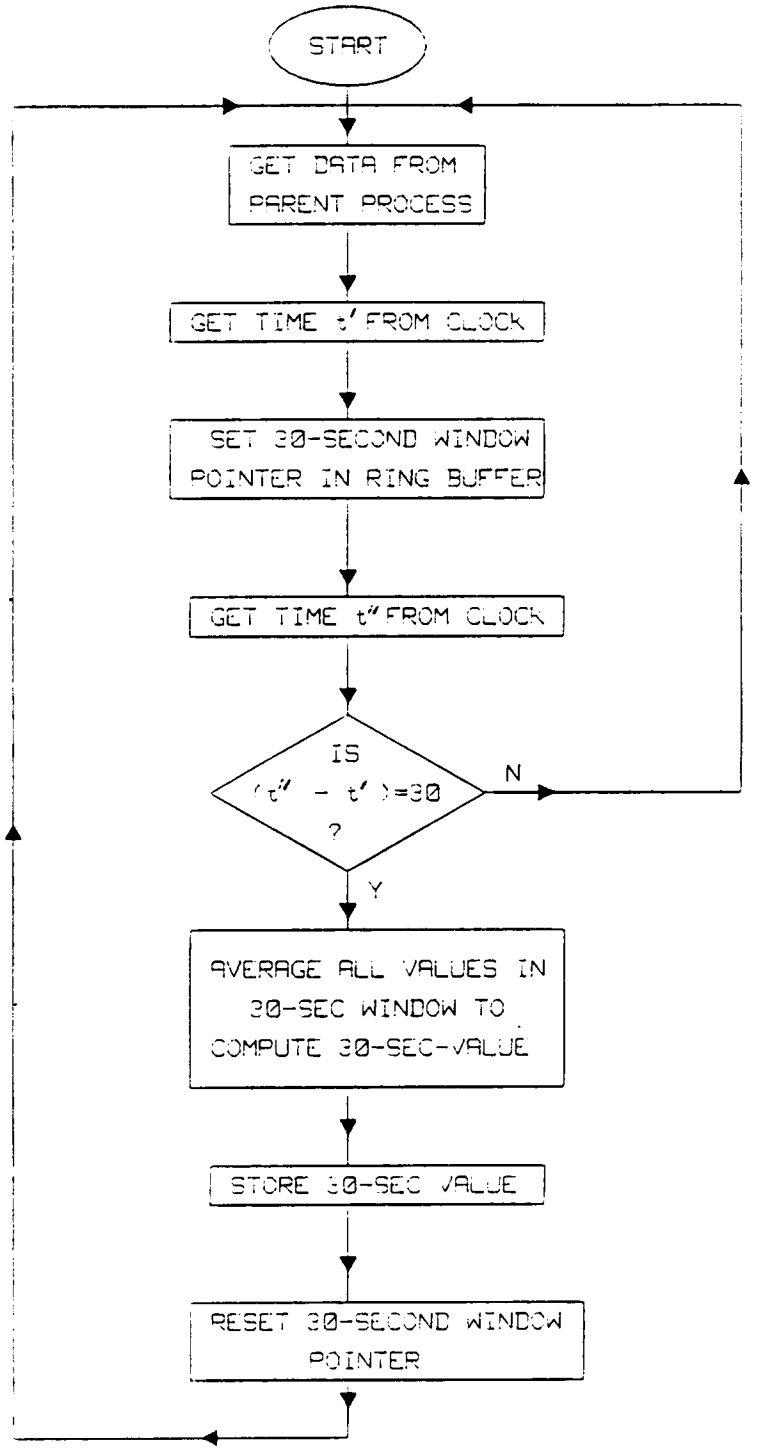

Figure 17. The flowchart for GET_LOAD.c code.

A.1 FLOWCHART FOR SOFTWARE ON THE 3B2.

The following two flowcharts listed are subroutines of the Data Acquisition software on the 3B2/300 supermicro. The Figure 17 on page 94 is the flowchart for subroutine GET_LOAD. This routine arranges the data in the ring buffer, and maintains the pointer to a 30-second data window in the ring buffer. Once the 30-second data value is created, it is written to a data pipe and the pointer is reset.

The second flowchart as shown in Figure 18 on page 96, outlines the technique developed for storing the 30-second, 15-minute and hourly data values in data files that scroll up after 24 lines of data. For details of the file format refer to "Results of Data Acquisition." on page 79 The files collect the archived data from data pipes and store them in the /ETC/ directory for any user to view.

The subroutines Penril, Filter and Hang_up are explained in details in the "Data Acquisition subsystem on the 3B2/300 supermicro." on page 55. The Penril and Hang_up routines are mainly for initialization of the I/O ports while the Filter code is the same as that employed on the IBM PC/XT. Hence, the flowcharts of these subroutines not been listed here.

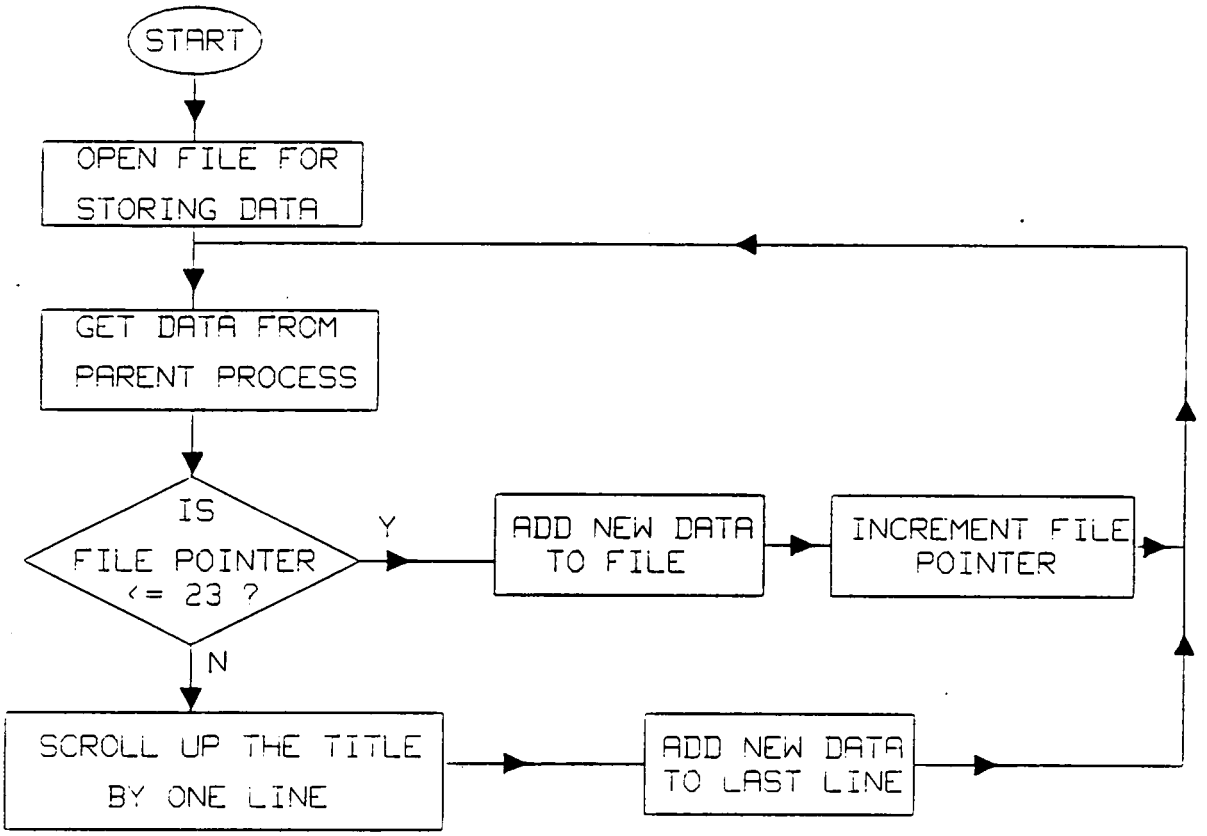

Figure 18. The flowchart for routines that format the data files.

APPENDIX B. PERIPHERAL SPECIFICATION

B.1 THE PENRIL MODEM

The AT&T ACU/Modem has been used as a peripheral for data acquisition on the IBM PC as well as on the 3B2. This modem can manually or automatically originate or answer telephone calls.

The ACU/Modem is always in one of two modes of operation: **OFF-line** or **ON-line**. When Off-line, the computer can communicate with the ACU in order to command and control modem operation. Listed in the table 1 are a few of the important commands that are used in the software designed in this project for communicating with the ACU/Modem. When the ACU modem is connected to a computer, in response to a carriage return <CR> from the host computer, it sends the following message

```
AT&T
1200 bps
>
```

The above response assumes baud rate set up for communication is 1200 BPS. This is set by the host computer before transfer of data. After the '>' prompt appears, the ACU is

ready to accept commands. The ACU abort any commands if it receives extraneous characters embedded in the command string. The command string sent to it by the host computer does not require a <CR> (carriage return) for termination. In this project, the phone number was stored in one of the store locations of the ACU and the ACU was made to dial automatically using the 'DIAL' command.

TABLE 1

COMMAND	DEFINITION	PROCEDURE
S0-S9	STORE NUMBER	Type "S", any location no. between 0-9, phone no. and <CR>
0-9	Dial (0-9)	Type "n" to dial a stored phone no. in location n (0-9)
R	Redial	Type "R"; ACU redials the last number dialed
K	Keyboard Dialing	Type "K", phone number and <CR> key

The command mode prompt for the Penril is the sign ">". In this mode the above commands and many others (see the INSTRUCTION GUIDE FOR THE ACU/MODEM) can be sent to the modem. This modem enters the command mode after the computer or the DTE configures the port connected to the modem with a correct set of communication parameters. The Table 2 lists a few of the important messeges sent by the modem in response to the commands sent to it.

TABLE 2

MESSAGE	DEFINITION
>	Prompt Character; ACU is waiting for command
ABORT	Character typed while ACU is processing a command or dialing a phone number
ERROR	Invalid command entry
Dialing ddd...	ACU displays number being dialed
No Dial Tone	ACU is waiting for dial tone and does not detect it
Busy	Number dialed is busy; ACU hangs up
No Ring	Call did not go through
OK	ACU is ONLINE in ORIGINATE mode

When ONLINE, the communication link is established with the remote modem through the phone line. The ACU "stands aside" and lets the DTE pass data through the modem to and from the remote computer.

When in ONLINE and ORIGINATE mode, as in the case of data acquisition, the modem will disconnect if one of the following occurs -

- Loss of DTR
- Loss of Carrier Detect
- The Quit Code is received from the DTE

In this project the DTR was made to drop by the central computer when required to disconnect the line. This was acheived by using the Hang_up command (refer to "HANG_UP.C" on page 73) which forces the DTR to go low on the port specified.

B.2 'CU' COMMAND IN BNU

As mentioned in "Data Acquisition subsystem on the 3B2/300 supermicro." on page 55, the **CU** command calls a remote computer and links the 3B2 computer to the called system.

B.2.1 Command Format

The **CU** command has the basic format:

CU [options] telephone number

The Table 3 lists the various options of the **CU** On the UNIX OS, when the '$' prompt appears, the 'CU' command can be typed as a shell command in the format shown above. As mentioned in Chapter 5, the 'CU' reads the dialing string from the 'DIALERS' file in the '/USR/LIB/UUCP ' directory. The options are specified on the command line and the shell command is terminated by a <CR> (carriage return). If the diagnostics option is specified, the 'CU' command responds with the diagnostics traces which is very useful for debugging.

TABLE 3

OPTION(S)	DESCRIPTION
-s speed	specifies baud rate; default is 300 bps
-l line	specifies the device port number to which the peripheral is connected. "line" is the port number.
-h	emulates local echo when in half-duplex mode
-t	used to indicate Auto Answer
-d	causes diagnostic traces to be printed
-e(-o)	designates even (odd) parity to be generated
-n	indicates that CU will prompt you for the telephone number instead of taking it from the command line

Once the connection is made and the computer is logged on to the remote system, the connection is disconnected only when the DTR is dropped. In this project, the Hang-Up command (refer to "HANG_UP.C" on page 73) drops the DTR from the 3B2, thus disconnecting the link. For further details about the usage of the **"CU"** command refer to the AT&T 3B2 COMPUTER BASIC NETWORKING UTILITIES GUIDE [6].

BIBLIOGRAPHY.

1. Instruction Guide for the KS-23095, L1 ACU/Modem by AT&T, 1984.

2. IBM Technical Reference Personal Computer XT, 1984.

3. UNIX System V Programming Guide.

4. S.R.Savitsky, " Real Time Microprocessor Systems ", Van Nostrand Reinhold Company, NY, 1985.

5. AT&T 3B2/300 Computer Owner/Operator Manual.

6. AT&T 3B2 Computer UNIX System V Release 2.0 Basic Networking Utilities Guide.

7. UNIX System V User's Guide.

8. Zaky, Hamacher and Vranesic, "Computer Organization ", Mcgraw-Hill, Japan, 1981.

9. Cermetek INFOMATE 212 PC User's Manual, 1984.

10. CP/M C-86 C-Compiler User's Manual, 1984.

11. AT&T 3B2 Computer UNIX System V Release 2.0 System Adminstration Utilities Guide.

12. UNIX System User's Manual.

13. S.G. Kochan and P.H. Wood, " Exploring the Unix system", Hayden Book Company, NJ, 1984

14. AT&T 3B2 Computer UNIX System V Release 2.0 Terminal Information Utilities Guide.

15. AT&T C-Programming Language Manual.

The two page vita has been removed from the scanned document. Page 1 of 2

The two page vita has been removed from the scanned document. Page 2 of 2